NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DETECTION AND CLASSIFICATION OF LOW PROBABILITY OF INTERCEPT RADAR SIGNALS USING PARALLEL FILTER ARRAYS AND HIGHER ORDER STATISTICS

by

Fernando L. Taboada

September 2002

Thesis Advisor:	Phillip E. Pace
Co-Advisor:	Herschel H. Loomis Jr.

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY	**2. REPORT DATE** September, 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Detection and Classification of LPI Radar Signals using Parallel Filter Arrays and Higher Order Statistics.			**5. FUNDING NUMBERS**
6. AUTHOR(S) Major Fernando L. Taboada			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT

The term *Low Probability of Intercept* (LPI) is that property of an emitter that because of its low power, wide bandwidth, frequency variability, or other design attributes, makes it difficult to be detected or identified by means of passive intercept devices such as radar warning, electronic support and electronic intelligence receivers. In order to detect LPI radar waveforms new signal processing techniques are required.

This thesis first develops a MATLAB® toolbox to generate important types of LPI waveforms based on frequency and phase modulation. The *power spectral density* and the *periodic ambiguity function* are examined for each waveform. These signals are then used to test a novel signal processing technique that detects the waveform parameters and classifies the intercepted signal in various degrees of noise. The technique is based on the use of parallel filter (sub-band) arrays and higher order statistics (third-order cumulant estimator). Each sub-band signal is treated individually and is followed by the third-order estimator in order to suppress any symmetrical noise that might be present. The significance of this technique is that it separates the LPI waveform in small frequency bands, providing a detailed time-frequency description of the unknown signal. Finally, the resulting output matrix is processed by a feature extraction routine to detect the waveform parameters. Identification of the signal is based on the modulation parameters detected.

14. SUBJECT TERMS Low probability of Intercept Radar Signals, Signal processing, Filter bank, Higher Order Statistics, Cumulants.			**15. NUMBER OF PAGES** 297
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DETECTION AND CLASSIFICATION OF LOW PROBABILITY OF INTERCEPT RADAR SIGNALS USING PARALLEL FILTER ARRAYS AND HIGHER ORDER STATISTICS

Fernando L. Taboada
Major, Venezuelan Army
B.S., Armed Forces University, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2002

Author:

Fernando L. Taboada

Approved by:

Phillip E. Pace
Thesis Advisor

Herschel H. Loomis Jr.
Co-Advisor

Dan C. Boger
Chairman, Information Sciences Department

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Low probability of intercept (LPI) is that property of an emitter that because of its low power, wide bandwidth, frequency variability, or other design attributes, makes it difficult to be detected or identified by means of passive intercept devices such as radar warning, electronic support and electronic intelligence receivers. In order to detect LPI radar waveforms new signal processing techniques are required.

This thesis first develops a MATLAB® toolbox to generate important types of LPI waveforms based on frequency and phase modulation. The power spectral density and the periodic ambiguity function are examined for each waveform. These signals are then used to test a novel signal processing technique that detects the waveform parameters and classifies the intercepted signal in various degrees of noise. The technique is based on the use of parallel filter (sub-band) arrays and higher order statistics (third-order cumulant estimator). Each sub-band signal is treated individually and is followed by the third-order estimator in order to suppress any symmetrical noise that might be present. The significance of this technique is that it separates the LPI waveform in small frequency bands, providing a detailed time-frequency description of the unknown signal. Finally, the resulting output matrix is processed by a feature extraction routine to detect the waveform parameters. Identification of the signal is based on the modulation parameters detected.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

ACKNOWLEDGMENTS

I want to thank Professor Phillip E. Pace for his guidance and support during this research effort. His knowledge and experience helped me overcome numerous obstacles and made this thesis a very rewarding experience.

I address special thanks to the Venezuelan Army, particularly to those people who made this course possible and have always been the military model to follow, General Guaicaipuro Lameda Montero and General Oswaldo Contreras Maza.

To my wife Renata and my son Andres, for all their love, support and understanding throughout our time in Monterey. You are the most important motivation in my life.

To all of You who have influenced my life and now can fell proud of me on earth or heaven: my mother Rosalba, my father Enrique, my brother Enrique and my three grandmothers Maruquita, Magda and Rosalba.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

In the past, many types of radar were designed to transmit short duration pulses having relatively high peak power, to reduce all the propagation losses of the electromagnetic waves and, at the same time, to guarantee a straightforward recovery of the reflected wave from the target in clutter. Later, for military applications, it became important to handle problems like chaff and jamming that decreased the capability of those initial radars limiting their use on the battlefield.

Today, radar designers are considering new waveforms that can provide the same capability of target detection but which are more difficult to be detected or intercepted; in other words, a Low Probability of Intercept (LPI) radar tries to provide detection of targets at longer ranges than intercept receivers can accomplish detection of the radar. The term LPI provides a collection of properties offering covertness to the radar signal making detection difficult with conventional receivers.

From the interceptor's point of view, the current and future use of LPI technology will encourage new approaches for the detection and interception of this type of radar signal. This thesis first develops a MATLAB® toolbox to generate important types of LPI waveforms based on frequency and phase modulation. The power spectral density and the periodic ambiguity function are examined for each waveform. Furthermore, this thesis documents an approach based on the use of parallel filter arrays and higher order statistics for the detection of most of the modulations employed today for radar applications. Additionally, this thesis provides a simple approach for the extraction of some of the parameters needed to identify and classify these signals.

Examined LPI waveforms include Frequency Modulation Continuous Wave (FMCW), several polyphase-coded CW waveforms such as Frank, P1, P2, P3 and P4, frequency hopping, and combined frequency hopping-phase coding. As pointed out in this text, this technique alone is not sufficient to process the multiplicity of available LPI waveforms, but the combined use of this technique with others, such as Wigner distribution, Cyclostationary processing, and Quadrature mirror filtering will provide the expected response.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OVERVIEW OF LPI RADAR SIGNALS

1. Low Probability of Intercept Radar Signals

In the past, many military types of radar were characterized by short duration pulses having relatively high peak power, with Radar Warning Receivers (RWR) and Electronic Support (ES) receivers being designed to detect those radars. Now, radar designers have considered new waveforms which are more difficult to intercept protecting them against Anti-Radiation Missiles (ARM) and reducing the detection range of RWRs and ES equipment. Today, many military users of radar are specifying Low Probability of Intercept (LPI) as an important technical and tactical requirement.

The LPI technique is based on the property of an emitter that due to its low power, wide bandwidth, frequency variability and other attributes makes radar difficult to intercept or identify by conventional passive intercept receiver devices. A LPI radar works to detect targets at a longer range than an intercept receiver. This concept is well-summarized in a statement "It tries to see and not be seen." This is a response to the increasing capability of modern intercept receivers to detect and locate radar emitters, possibly leading rapidly to an electronic attack or the physical destruction of the radar by guided munitions or Anti-Radiation Missiles.

The proliferation of radar, altimeters, tactical airborne targeting, surveillance and navigation devices employing LPI capabilities has demonstrated that power spectral analysis is useless when intercepting these signals; therefore, a more sophisticated signal processing must extract the necessary parameters of the waveform to create a proper electronic response. Table 1 shows some of the LPI systems currently in use as well as their developers and the waveform they employ (if known). Table 2 presents various definitions frequently used in LPI systems.

Developer	System	Technique used	LPI Use
Honeywell	HG9550	Frequency agility	Radar Altimeter
Navair	GRA-2000	-	Tri-service radar altimeter
NavCom Defense Electronics	AN/APN-232	FMCW	Combined Altitude Radar
Thompson CSF	AHV-2100	-	Radar Altimeter
BAE	AD1990	Frequency hopping	Radar altimeter
Saab Bofors	Pilot Mk 1,2,3	Fast frequency hopping	Surveillance, navigation
Signaal's	Scout	FMCW	Surveillance, navigation
Textron Systems	AN/SPN-46		Precision approach, landing
Signaal's	Smart-L	-	Surveillance
TI	AN/APS-147	Frequency agility	Enhanced search, target designation
Sierra Nevada	TALS	-	Tactical automatic landing system
Ericsson	Eagle	-	Fire control
Northrop Grumman	AN/APG-77	Frequency agility	Multi-mode tactical radar for F-22
Raytheon	AN/APG-70	Frequency agility	Multi-mode tactical radar for F-15E
TI	LANTIRN	-	Terrain following radar F-16C/D, F-15E, F-14
Raytheon	AN/APG-181	-	Multi-mode radar for B-2
Chinese	JY-17A	-	Battlefield surveillance radar
Raytheon	MRSR	-	Target acquisition and tracking
Saab Dynamics	RBS-15MR	-	Radar guided air-to-surface missile

Table 1 LPI radar systems (from [1]).

Terms	Definition
Coherent Radar	Transmitted signal has a constant phase relationship to an oscillator in the transmitter
Frequency-Agile Radar	Pulse or group of pulses are transmitted at different frequencies
LPI Radar	A radar with parameters that make it difficult for an ES receiver to correctly identify the radar type
Quiet Radar	A radar that detects a target at the same range that the target can detect the radar's signal.
Random-Signal Radar	A radar which uses a waveform that is truly random (e.g., noise)
Poly-Phase-Coded Continuous-wave radar	A radar that has a pseudo-random phase-coded modulation on a transmitted continuous-wave signal

Table 2 Definition of LPI radar (from [1]).

2. Characteristics of LPI Radar Signals

A LPI radar design must focus on the ability to defeat all the external threats that can lead to a precise identification of the system. Therefore, the following systems must be carefully designed to achieve the desired capability:

- Security of the matched filter
- Minimized signal PSD
- Randomized radar parameters
- Wideband operation
- LPI antenna design
- Power management

To make the radar covert, the knowledge of a matched filter must be denied to unintended observers. This implies that the system requires a large selection of available waveforms with poly-phase coding supplying the necessary diversity.

A LPI radar requires wideband signal modulations that reduce the signal's detectability. Wideband modulations spread the signal's energy in frequency, so that the frequency spectrum of the transmitted signal is wider than what is required to carry the signal's information (Information bandwidth). Spreading the signal energy reduces the signal-strength-per-information bandwidth. Since the noise in a receiver is a function of its bandwidth, the SNR in any receiver attempting to receive and process the signal will be greatly reduced by the signal spreading.

There are three ways in which modulation is used to spread the signal in frequency:

- Periodically changing the frequency
- Sweeping the signal frequency at a high rate, or chirping; and
- Modulating the signal with a high rate digital signal, or direct sequence-spectrum spreading.

Included in these categories are many wideband modulation techniques available to provide secure LPI waveforms:

- Frequency Modulation
- Linear FM (Chirp)
- Non-Linear FM
- Frequency Modulation Continuous Wave (FMCW)
- Costas Array, frequency hopping
- Phase modulation (bi-phase coding, polyphase coding)
- Combined phase shift keying, frequency shift keying (PSK,FSK)
- Pseudo-noise modulation
- Polarization modulation

Minimizing the radiated spectral density is another obtainable LPI requirement. If the code modulation is restricted to codes having two-level autocorrelation functions, the root-mean-square (rms) side lobe values are likewise reduced.

The most important antenna characteristic for reducing the possibility of an intercept receiver detecting the radio frequency (RF) emissions is a low-side-lobe-transmit pattern. Another antenna technique relates to the scan pattern, which must be precisely controlled to limit the intercept receiving time making it short and irregular. Antenna techniques can also achieve an LPI capability by using multiple simultaneous receive and transmit antenna beams to increase the target dwell time without compromising the target revisit time. Following the same criteria, a single wide beamwidth transmit antenna and many simultaneous receiving beams could also be employed.

One important characteristic for LPI radar is the ability to manage the transmitted power (limiting its emission) in order to keep the target's SNR constant. Using wideband continuous waves (CW) emissions, it is only necessary to transmit low energy to detect targets instead of tens of kilowatts required for pulse radar with similar performance, as shown in Figure 1 .

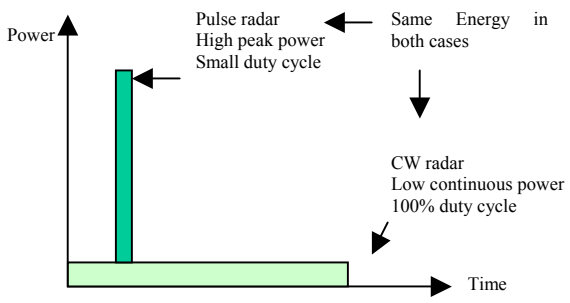


Figure 1 Comparison of a Pulsed Radar and a CW Radar (from [1]).

The high peak power transmitted by the pulsed radar can easily be detected by electronic support (ES) receivers. On the contrary, CW signals have a peak-to-average power ratio of one (100% duty cycle) transmitting very low power while maintaining the same energy profile.

The sensitivity factor is a crucial parameter that must be evaluated to succeed in the design of LPI radar. As shown in Figure 2 , sensitivity is a function of the bandwidth, noise figure and required SNR. The thermal noise is based on the formula *KTB* where *T* is the temperature in Kelvin, *K* is the Boltzmann's constant and *B* represents the bandwidth. The sensitivity in dBm is the sum of the thermal noise (in dBm), noise figure (in dB), and required signal-to-noise-ratio (in dB). If we set the value of the SNR to 13 dB, then *KTB* is usually taken as

$$KTB = -114dBm + 10\log(B)$$

where *KTB* is the thermal noise in dBm and *B* is the bandwidth on Hz.

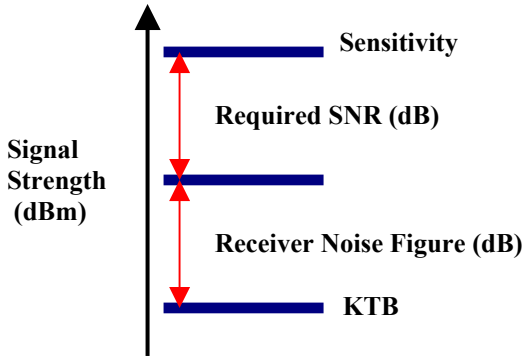


Figure 2 Receiver Sensitivity (from [2]).

Another useful factor to consider in the context of LPI radar signals is processing gain. Processing gain has the effect of narrowing the effective bandwidth of the radar receiver by taking advantage of some aspects of the signal modulation. The advantage comes because the radar receiver achieves a processing gain while the hostile intercept receiver cannot. An LPI radar achieves bandwidth advantage over an intercept receiver, because the radar has a matched filter to its own signal. In contrast, the intercept receiver must accept a wide range signals and must typically make detailed parametric measurements to identify the type of signal it is receiving.

Coherent detection is an additional factor that help an LPI radar signal from being intercepted. Existing Electronic Warfare (EW) receivers cannot achieve coherent detection of a radar signal unless they know details of the signal. When the signal modulation is random, this property becomes even more effective. Using a true noise to modulate a radar signal is a good illustration of this characteristic. Radars using true noise modulation are called random-signal radars (RSR). This kind of radar uses a technique to correlate the returning signal with a delayed sample of the transmitted signal. The amount of delay necessary to peak the correlation determines the range to the target. Since the transmitted signal is completely random, the intercepting receiver has no reference for correlating the received signal.

The use of frequency bands strategically located in the atmospheric absorption region creates difficulty for an intercepting receiver to detect the emissions. In Figure 3 , at least five different peaks are observable when covering the emission of LPI radar. This is an important constraint since radar depends exclusively on the energy placed on the target and the energy reflected from it. Based on this fact of physics, a LPI radar must radiate enough power to avoid a complete absorption of the signal but not enough to be detected by intercept receivers. Due to this limitation, this technique is only useful in radar with a short detection range.

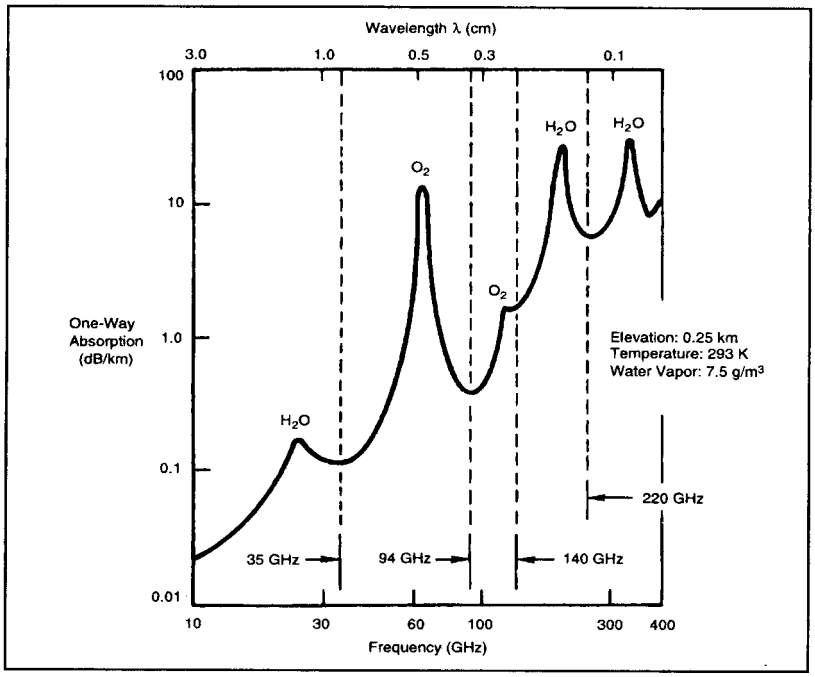


Figure 3 Regions of minimum and maximum atmospheric absorption for millimeter wave spectrum (from [3]).

3. LPI Intercept Receivers

Due to the nature of the new LPI threat, modern intercept receivers are becoming increasingly ineffective. Detection and interception of these LPI signals require sophisticated

digital receivers, that use time frequency signal processing and correlation techniques to collect signal data, do the analysis and generate an electronic attack (jamming).

> Menahem Oren, general manager of ELISRA Electronic System (Bene Beraq, Israel), states, "LPI modulations cannot be properly processed with 'snap shots' of data. These signals will require the collection of continuous streams of data. All of the current signal can be collected and processed, but only with digital receivers can detect LPI signals." [2]

New solutions to the detection of LPI radar signals have revealed new challenges to overcome. These are some of the new problems that intercept receiver designers must face:

- Differential Doppler provides an extremely accurate technique for emitter location. The measurements are based on the frequency shifts caused by the velocity of the aircraft making the measurements. This element causes considerable problem for airborne emitters.
- Increasing the measurement fidelity means that more beams of data are produced. These require more computing speed and more memory. Managing processing speed is not a problem with the current digital capabilities but carrying enormous amounts of data is still a problem.
- Increasing sensitivity of the receiver allows detecting sidelobes of the emitter but, at the same time, obligates the receiver to process a significantly large number of signals.

In addition to these challenges, new signal-detection and feature-extraction systems are needed to effectively analyze these new waveforms in today's complex signal environment. Developing new Electronic Attack (EA) techniques and evaluating the performance against LPI radar systems requires new theoretical approaches and a good deal of simulation and modeling.

Time-frequency data analysis can be performed using complex instrumentation through computer analysis. Computer algorithms are currently being developed to analyze and graphically display the results of the data for user interpretation. Improvements are being considered to provide representations beyond the conventional use of the Fast Fourier Transform (FFT). The use of Higher Order Statistics (HOS) and parallel filter arrays along with the extraction of the most important features provide an accurate analysis and interpretation of unknown signals in real time. The documentation of this technique is the objective of the present work.

This thesis documents the use of parallel filter arrays and HOS as an effective technical approach for detecting and classifying LPI radar signals where the waveform of the signal is unknown. The HOS processing is one time-frequency approach to the detection of LPI signals (Figure 4). The objective of parallel filter arrays is to separate the input signal into small frequency bands, providing a complete time-frequency description of the unknown signal. Then, each sub-band signal is treated individually by a third-order estimator in order to suppress the noise and preserve the phase of the signal during the correlation process. Finally, the resulting matrix is entered into a feature extraction module whose resulting characteristics from the signal are used to determine what type of modulation was detected.

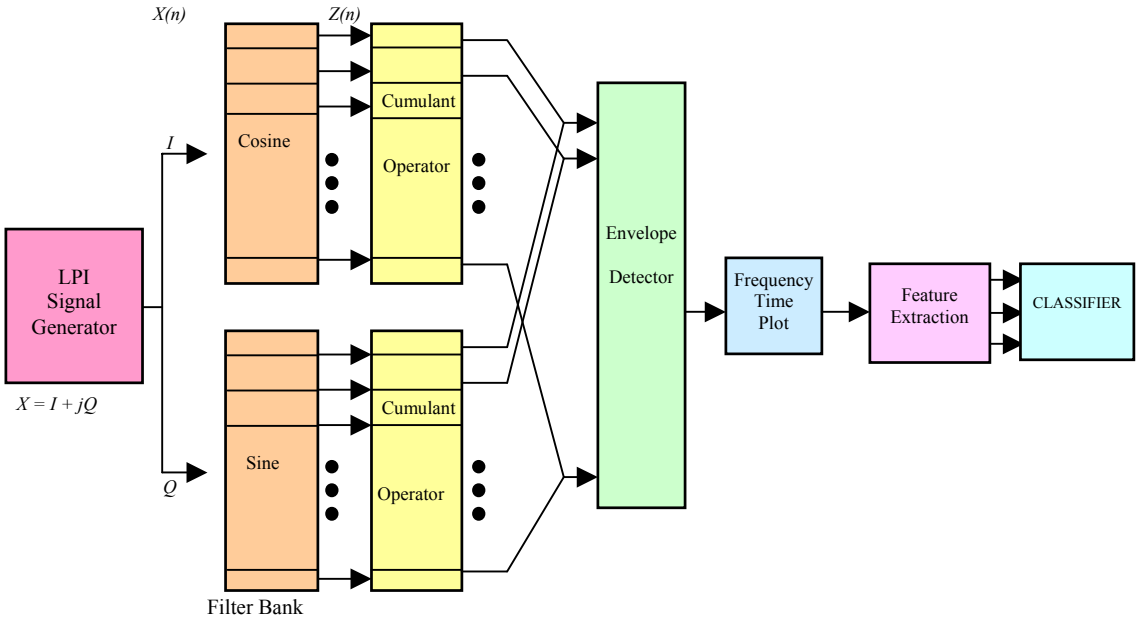


Figure 4 Overview of the parallel filtering and HOS.

B. PRINCIPAL CONTRIBUTIONS

The objective of the research described in this thesis is to design a signal processing scheme capable of detecting and classifying LPI radar signals based on the combined use of parallel filter arrays and HOS.[2]

Higher Order Statistics is a field of statistical signal processing which has become very popular in the last 15 years. This field makes use of information beyond that usually

used in traditional signal-processing measures (such as the power spectrum and the correlation function). This additional information can be used to incur better estimates of parameters in noisy situations.

This research proves that the combined use of parallel filter arrays and HOS is an important tool, providing a complete time-frequency analysis of the unknown signal, while also suppressing Gaussian noise and extracting information due to deviations from Gaussianity. In addition, the analysis based on HOS preserves the true phase character of signals. As proven, this proposed method alone is not sufficient to process the multiplicity of available LPI waveforms; however, the combined use of this technique with others, such as Wigner distribution, Cyclostationary analysis or Quadrature filtering will provide the necessary signal processing for tomorrow's LPI intercept receivers.

C. THESIS OUTLINE

Chapter II introduces the reader to the MATLAB®* toolbox for generating LPI signals. The lack of data from real radar signals motivated the design of a collection of mathematical models to generate signals with LPI properties, to be used later as inputs to the proposed detector and classifier. This generator by itself represents an important contribution for future research in this field.

Chapter III presents the complete design of parallel filter arrays and the implementation of the higher order estimators. It thoroughly explains the design of a uniform filter bank and the implementation of HOS.

Chapter IV illustrates the method used for feature extraction from the signals and the performance of the classifier based on the results. Results are shown and analyzed in detail along the parameters of the inputs signal for comparison.

*MATLAB® is a language that integrates mathematical computing and visualization to provide a flexible environment for technical computing. The open architecture makes it easy to use MATLAB® and its products to explore data, create algorithms, and create customs tools that provide early insights and competitive advantages.

Chapter V summarizes the results of this thesis and also makes concluding remarks and recommendations. The chapters are followed by Appendix A and B containing the MATLAB® m-files used in the simulation and modeling of the LPI radar-intercept receiver study.

In Chapter I, this document introduced the concept of LPI signals, showing their most significant characteristics, uses and the necessity for an innovative signal processing to overcome the new threat. Chapter II illustrates the most important modulations employed to obtain LPI characteristics, such as BPSK, FMCW and polyphase codes. The description of an LPI signal generator toolbox is presented as well as a tutorial to guide its use.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LOW PROBABILITY OF INTERCEPT EMITTERS AND THEIR SPECTRAL PROPERTIES

A. BINARY PHASE SHIFT-KEYING

Binary Phase Shift-Keying (BPSK) is modulation technique that has proven to be extremely effective in communication and radar systems. Even though BPSK is not a technique employed in LPI radar modulation, the technique is an excellent test signal in evaluating the performance of the proposed signal processing.

In BPSK modulation, the phase of the frequency carrier is shifted 180 degrees in accordance with a digital bit stream. The digital coding scheme used in this implementation is called Non-Return-to-Zero (NRZ-M). A "one" causes a phase-transition, and a "zero" does not produce a transition. Figure 5 shows a basic block diagram of the transmitter design. The signal $x(t)$ is a continuous wave (CW) sinusoid. After sampling at the Nyquist rate, the modulated signal is created by adding a n-bit Barker code. This code has been used widely because of its low-side lobes at zero Doppler. Once the signal has been modulated, white Gaussian noise is added with the desired Signal-to-Noise Ratio (SNR). [4]

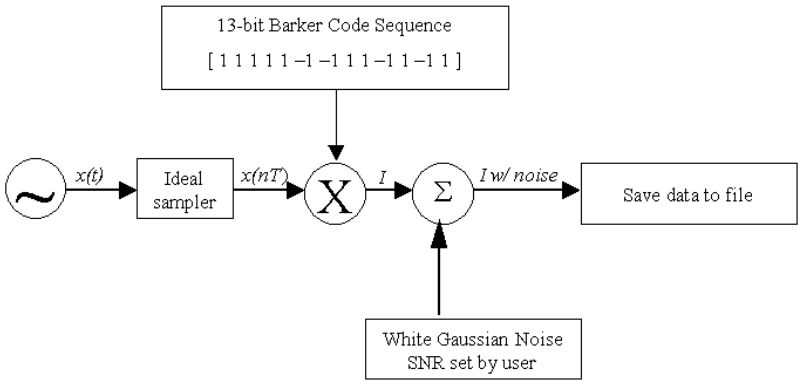


Figure 5 BPSK implemetation block diagram.

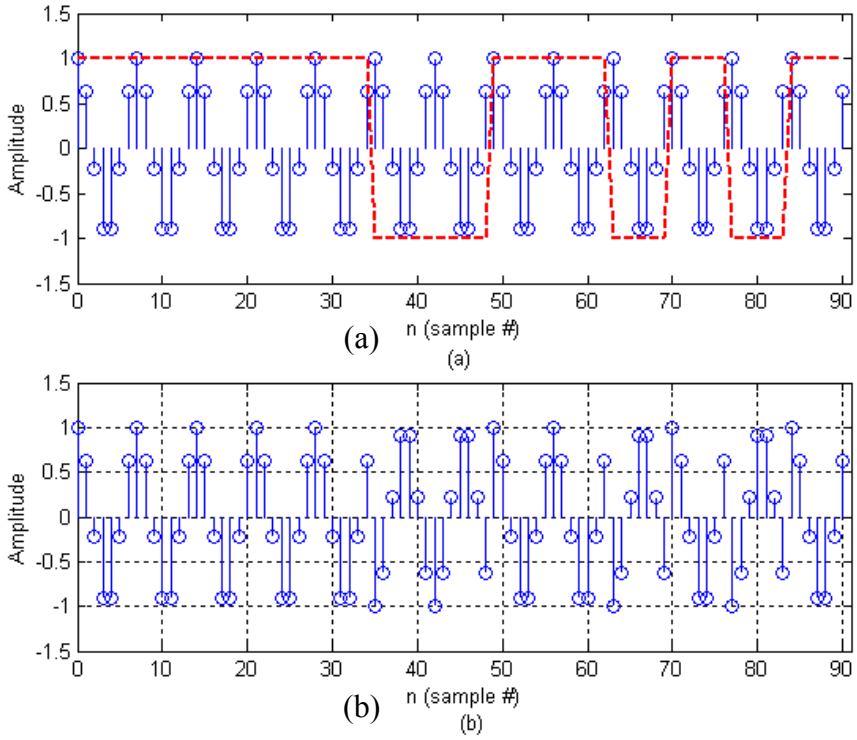


Figure 6 (a) Sampled signal and modulating signal in red (b) modulated signal for a 13-bit Barker code BPSK signal.

Figure 6 (a) shows the sampled signal and modulating signal and Figure 6 (b) presents the modulated signal for a 13-bit Barker code. The dashed red lines in Figure 6 (a) represent the modulating signal waveform. The number of periods of the carrier frequency per Barker Bit is equal to one, meaning that one full period of the sampled signal fits within one bit of the 13-bit Barker code. The first five bits of the Barker code are +1 and the next 2 bits are -1, so we see five full periods under the first +1 portion of the modulating waveform, 2 full periods under the -1 portion of the modulating waveform, and so forth. In this figure, a complete 13-bit Barker code is represented. Table 3 shows the Barker code sequence for 7, 11 and 13 bits.

Number of bits	Barker Code
7	+ + + - - + -
11	+ + + - - - + - - + -
13	+ + + + + - - + + - + - +

Table 3 Barker Code for 7, 11 and 13 bits.

Figure 7 illustrates the Power Spectral Density (PSD) of a 13-bit BPSK signal with carrier frequency equal to 1000 Hz, sampling frequency equal to 7000 Hz and 1 cycle per bit. Figure 8 shows the sampled signal in the time domain modulated by a 13-bit Barker code (a)without noise and (b) with SNR = 0 dB.

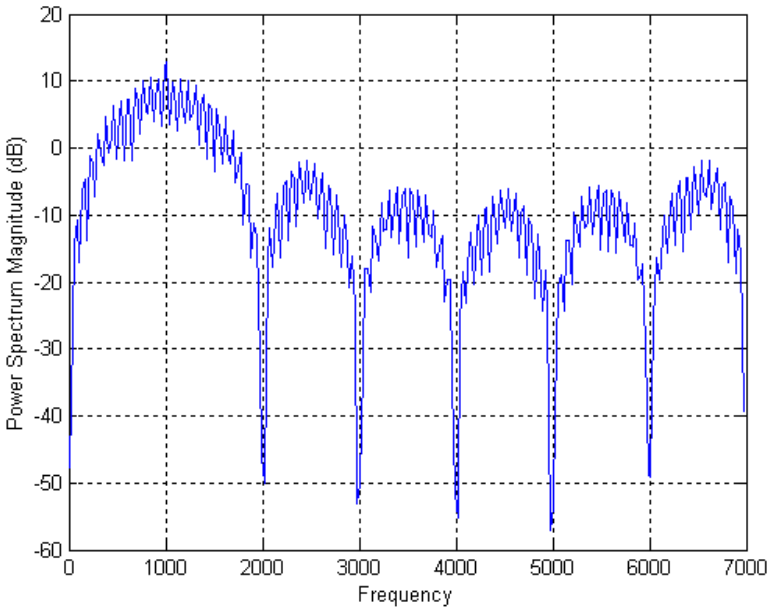


Figure 7 PSD of a BPSK signal modulated with 13-bit Barker code.

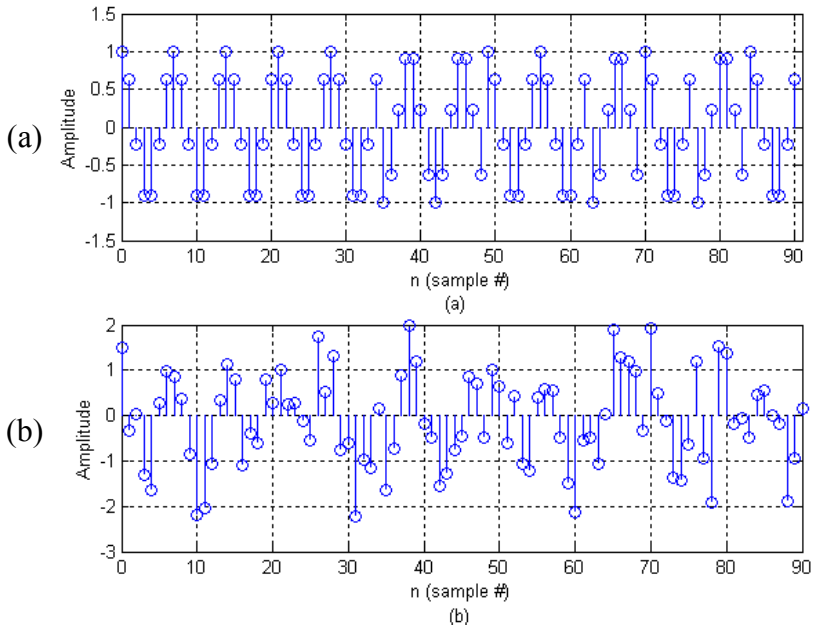


Figure 8 Sampled 13-bit BPSK signal (a) without noise and (b) with SNR = 0 dB.

Figure 10(a) shows the contour plot of the Periodic Ambiguity Function (PAF) of the 13-bit Barker CW signal [5] . This plot describes the response of the correlation receiver to the signal modulated by the 13-bit Barker code. The response is a function of both the delay and Doppler. Figure 10(b) presents the cut along the 0 Doppler axis corresponding to the perfect periodic autocorrelation. For a Barker code, the sidelobes peak level can be obtain by *1/N* where *N* is the number of bits processed. Figure 10(b) shows the cut along the 0 delay axis.

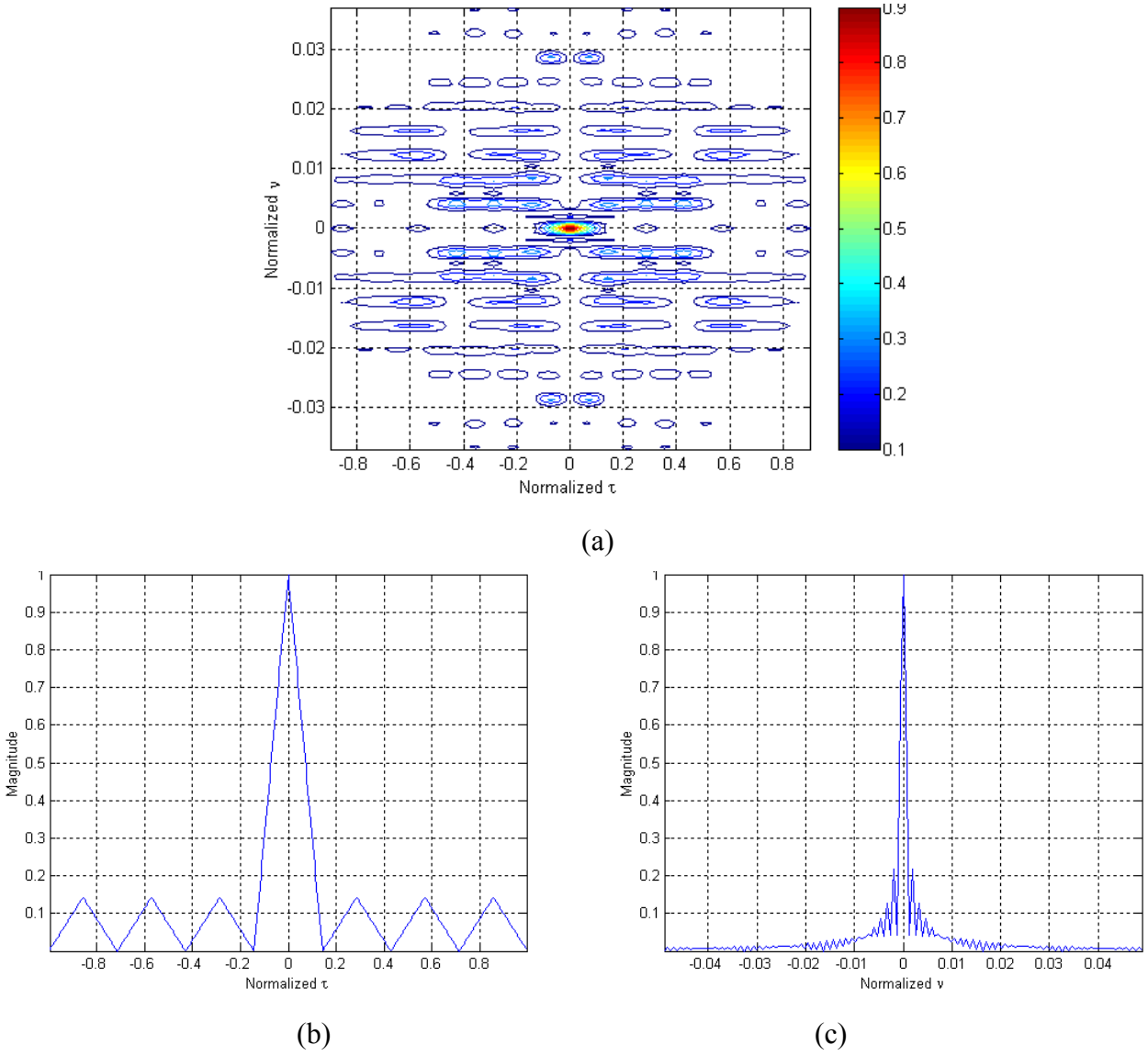


Figure 9 (a) Contour plot of the PAF for a BPSK signal modulated with 13-bit Barker code. (b) Cut along the 0 Doppler and (c) 0 delay axis.

B. FREQUENCY MODULATED CONTINUOUS WAVE

Linear frequency modulation is a LPI radar technique and is readily compatible with solid-state transmitters. The most popular modulation is the triangular modulation of a Frequency Modulated Continuous Wave (FMCW). The linear FMCW emitter uses a continuous 100 % duty-cycle waveform so that both the target range and the Doppler information can be measured unambiguously while maintaining a low probability of intercept [1]. The FMCW waveform shows excellent characteristics for the best use of the output power available from solid-state devices. This waveform is easier to implement than phase code modulation as long as there is not a strict demand on the linearity over the modulation bandwidth.

The triangular modulation consists of two linear frequency modulation sections with positive and negative slopes. With this configuration, the range and Doppler frequency of the detected target can be extracted unambiguously by taking the sum and the difference of the two beat frequencies. These characteristics are shown in Figure 10 . [1]

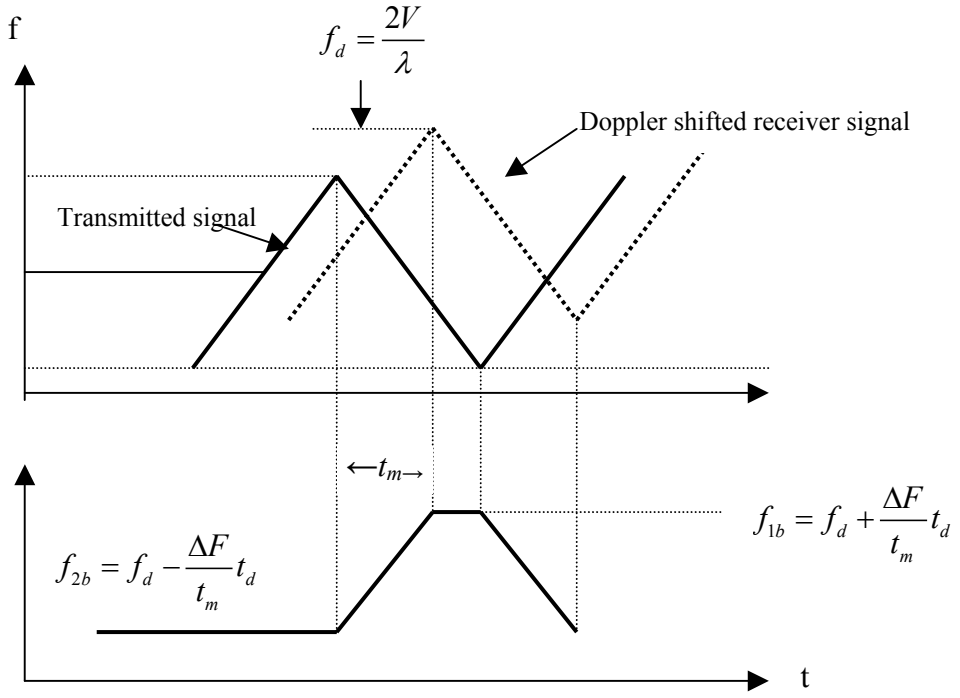


Figure 10 Linear frequency modulated triangular waveform and the Doppler shifted return signal.

The frequency of the transmitted signal for the first section is

$$f_1(t) = f_0 - \frac{\Delta F}{2} + \frac{\Delta F}{t_m} t \tag{2.1.1}$$

for $0 < t < t_m$ and zero elsewhere. Here f_o is the RF carrier, ΔF is the transmitted modulation bandwidth, and t_m is the modulation period.

The phase of the transmitted RF signal is

$$\phi_1(t) = 2\pi \int_0^t f_1(x)dx \tag{2.1.2}$$

Assuming that $\phi_o = 0$ at $t=0$ then

$$\phi_1(t) = 2\pi \left[\left(f_o - \frac{\Delta F}{2} \right) t + \frac{\Delta F}{2t_m} t^2 \right]$$

for $0 < t < t_m$. The transmit signal is given by

$$s_1(t) = a_o \sin 2\pi \left[\left(f_o - \frac{\Delta F}{2} \right) t + \frac{\Delta F}{2t_m} t^2 \right] \tag{2.1.3}$$

The frequency of the transmitted waveform of the second section is

$$f_2(t) = f_0 + \frac{\Delta F}{2} - \frac{\Delta F}{t_m} t$$

for $0 < t < t_m$. The transmitted base band signal is given by

$$s_2(t) = a_o \sin 2\pi \left[\left(f_o + \frac{\Delta F}{2} \right) t - \frac{\Delta F}{2t_m} t^2 \right] \tag{2.1.4}$$

Figure 11 illustrates the triangular modulation signal for a FMCW signal with a modulation bandwidth of 250 Hz, modulation period of 50 ms and carrier frequency of 1000 Hz.

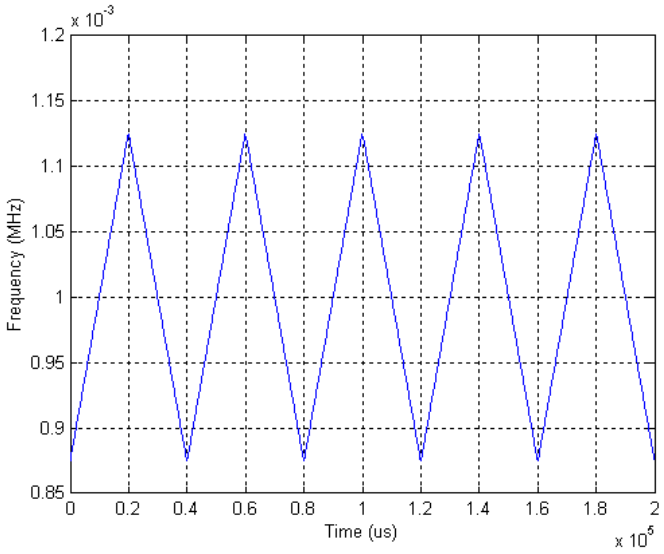


Figure 11 Triangular modulating signal for a FMCW.

Figure 13 shows the PSD of the triangular FMCW signal described above. Only the carrier frequency and modulation bandwidth are easily identified.

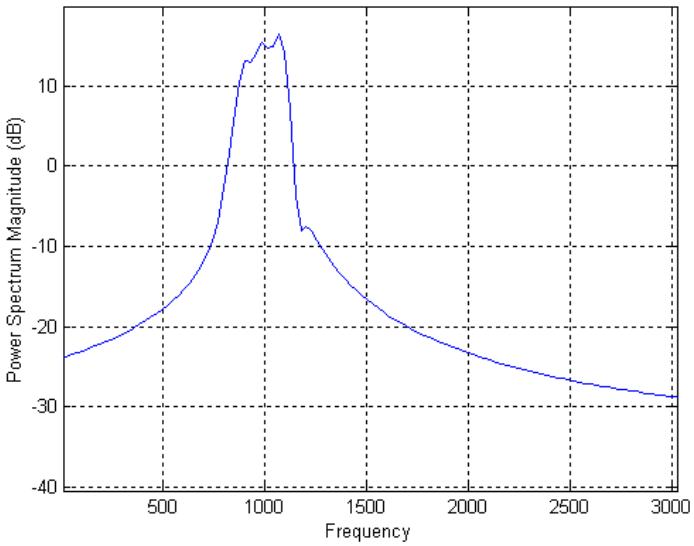
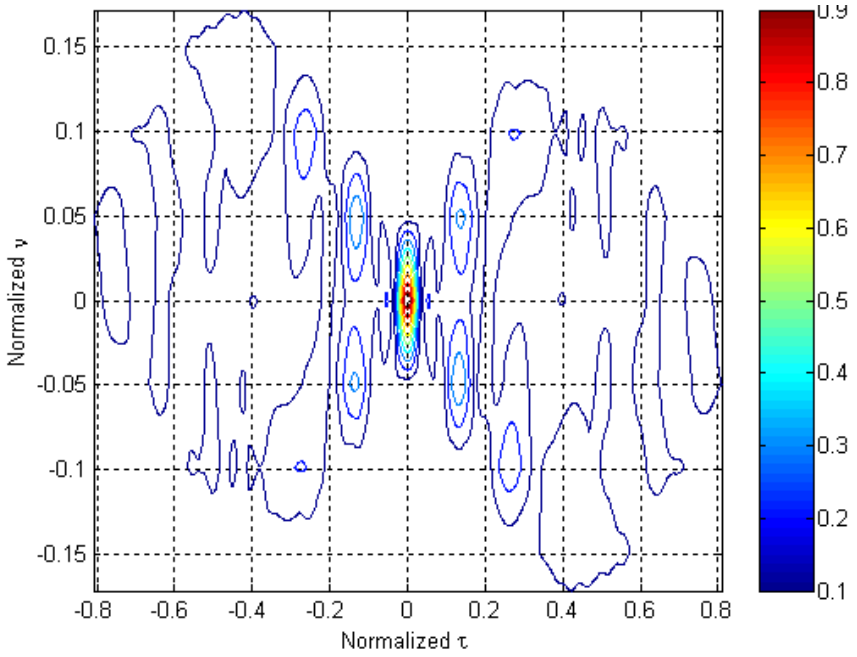
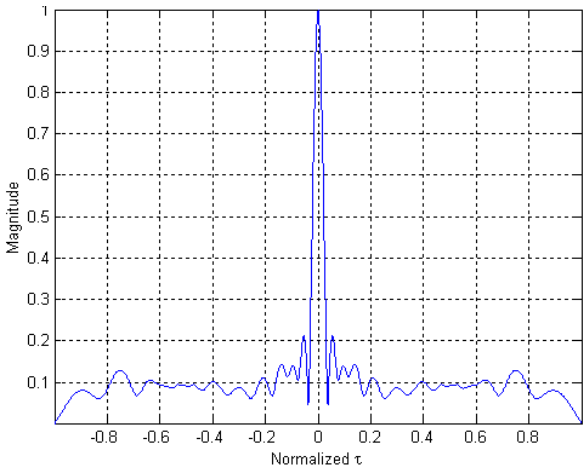


Figure 12 PSD of the FMCW signal described in Figure 11.

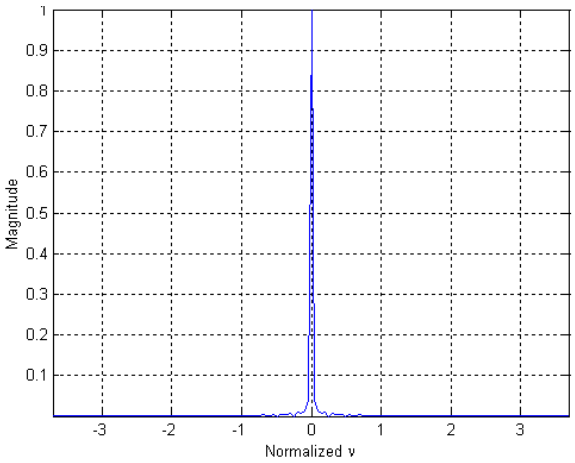
Figure 13 (a) illustrates the PAF of the signal. Additionally, Figure 13 (b) and (c) provides the cuts along the 0 Doppler axis and 0 delay axis.



(a)



(b)



(c)

Figure 13 FMCW signal (a) PAF, (b) Cut along 0 Doppler and (c) Cut along 0 delay.

C. FRANK CODE

The Frank code belongs to the family of polyphase codes. This code has been successfully used in implementing LPI radar signals. A Frank-coded waveform consists of a constant amplitude signal whose carrier frequency is modulated by the phases of the Frank code.

For each frequency or section of the step chirp, a phase group consisting of N phases samples is obtained and the total number of code phases is N^2, which is equal to the pulse compression ratio. If a local oscillator is at the start of the sweep of a step approximation to a linear frequency waveform, the first N samples of the polyphase code are 0 phase. The phase increments within the four phase groups are 0º, 90º, 180º and 270º. However the phases at the last group are ambiguous (>180º) and appear as -90º phase steps, or as the conjugate of the first group of phases.

The representation of a Frank-coded signal, where i is the number of samples and j is the number of frequency, the phase of the ith sample of the jth frequency is given by the following equation:

$$\phi_{i,j} = \frac{2\pi}{N}(i-1)(j-1) \tag{2.1.5}$$

where i=1, 2, ..., N and j=1, 2, ..., N. Each element of the Frank code is t seconds long, which is approximately equal to the reciprocal of the waveform 3 dB bandwidth. The phases of the Frank code may be generated for transmission by multiplying the elements of the matrix

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 2 & \dots & (N-1) \\ 0 & 2 & 4 & \dots & 2(N-1) \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & (N-1) & 2(N-1) & \dots & (N-1)^2 \end{bmatrix} \tag{2.1.6}$$

by the phase $2\pi / N$ and by transmitting the phases of row 1 followed by row 2 etc. The gross shape of the spectrum of an ideally generated Frank-coded waveform is approximately given by $\frac{\sin(\pi f / B)}{(\pi f / B)}$ where B is $1/\tau$, centered on the carrier frequency. In radar applications, transmitters are operated in saturation; therefore, the abrupt phase transition can be made from one code element to the next. Figure 14 shows the phase shift of a Frank-coded signal for N^2=16. The components of the matrix are plotted by rows.

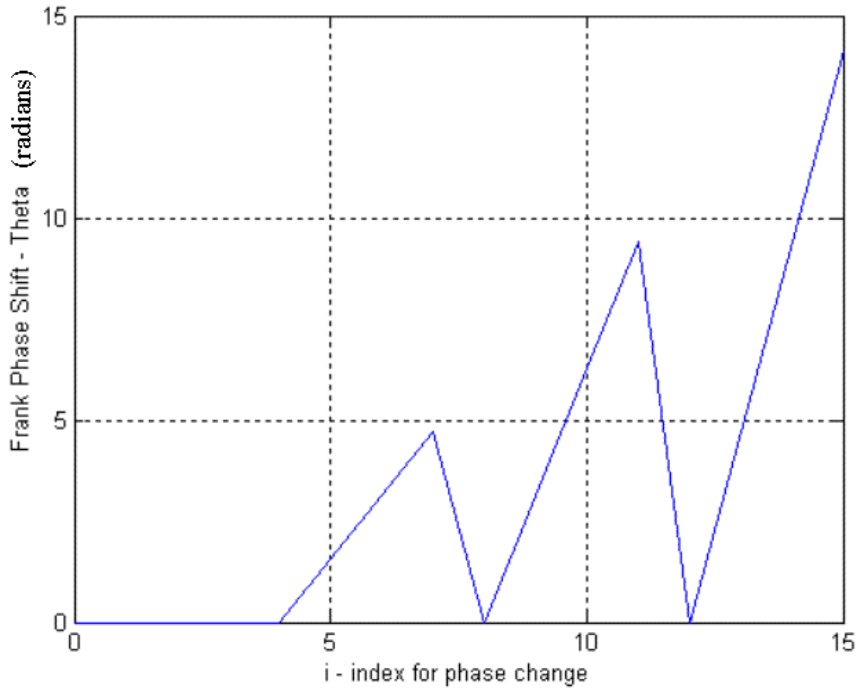


Figure 14 Phase shift in radians versus index in the matrix for N=4.

Figure 15 provides the PSD of the Frank-coded signal described above. One cycle per phase used in the generation of this signal. Some values can be identified in the plot, such as carrier frequency equal to 1000 Hz and bandwidth of 1000 Hz.. Figure 16 shows a time domain section of the signal.

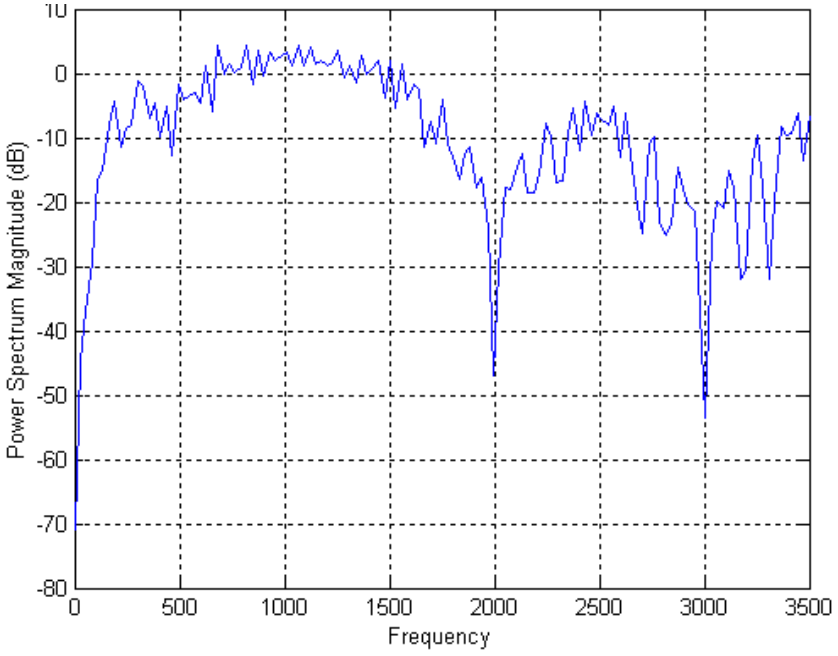


Figure 15 PSD for a Frank-coded signal with N=4.

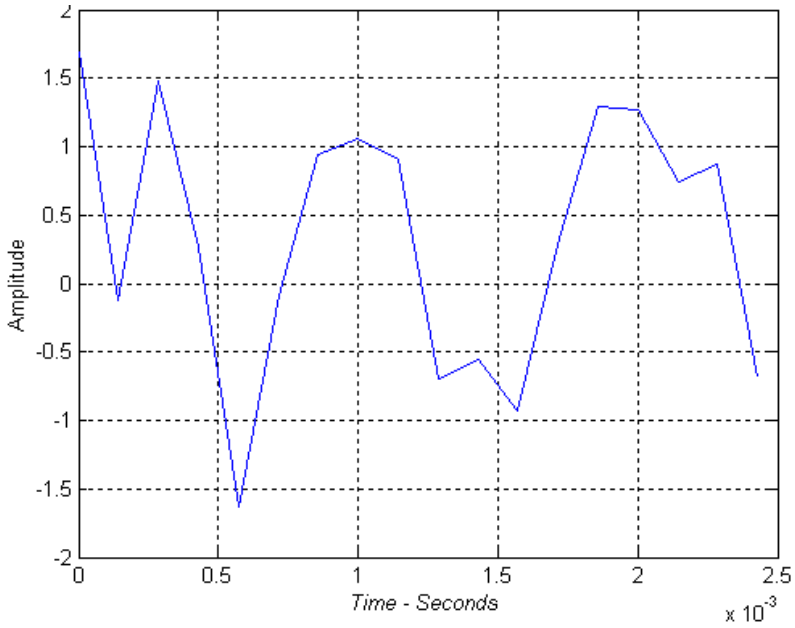


Figure 16 Time domain plot for a Frank-coded signal with N=4.

Figure 17 provides the contour plot of the PAF of the Frank-coded signal with N=4. Because N code groups, each having N code elements, add to form the match-point peak, the peak squared to peak sidelobe squared ratio will be

$$\frac{(peak)^2}{(peak\ sidelobes)^2} = \frac{(N^2)^2}{\left(\frac{N}{\pi}\right)^2} = \pi^2 N^2 \qquad (2.1.7)$$

for large N, the Frank polyphase code produces mean square sidelobes that are down on the order of $10\pi^2 N$ from the match-point peak squared. The cuts along 0 Doppler and the 0 delay axis are provided in Figure 18 for N=4.

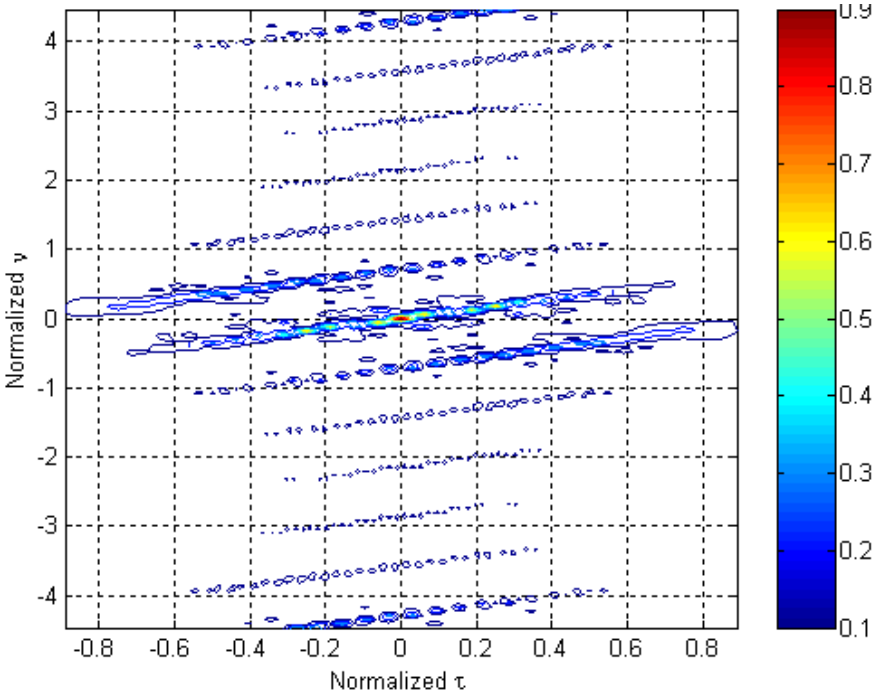


Figure 17 Contour plot of the PAF for a Frank-coded signal with N=4

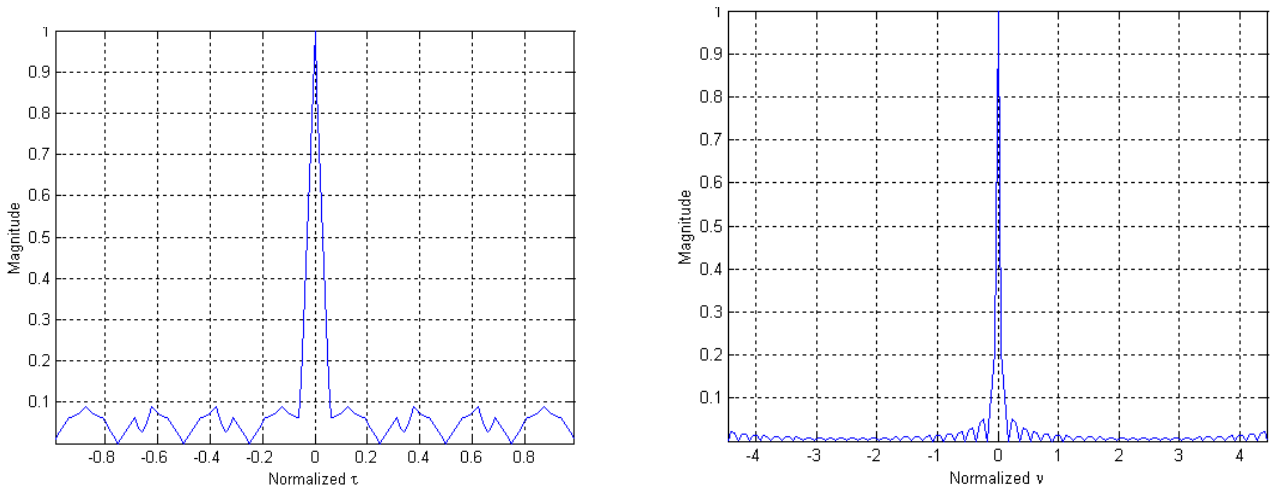


Figure 18 Cuts along the 0 Doppler and 0 delay axis.

D. P1 CODE

By changing the synchronous oscillator frequency, different phase codes can be generated with equal amplitudes but with different phases. By placing the synchronous oscillator at the center frequency of the step chirp IF waveform and by sampling the base band waveform at the Nyquist rate, the polyphase code called P1 may be obtained. The P1 code and the Frank code consist of same number N^2 elements. [6]

If i is the number of the samples in a given frequency and j is the number of the frequency, the phase of the ith sample of the jth frequency is given by the equation:

$$\phi_{i,j} = \frac{-\pi}{N}[N-(2j-1)][(j-1)N+(i-1)] \qquad (2.1.8)$$

where $i = 1,2,\ldots,N$ and $j = 1,2,\ldots,N$ The PAF for P1 code for N odd is identical to the Frank code. Figure 19 illustrates the phase shift of a P1-coded signal with N=8.

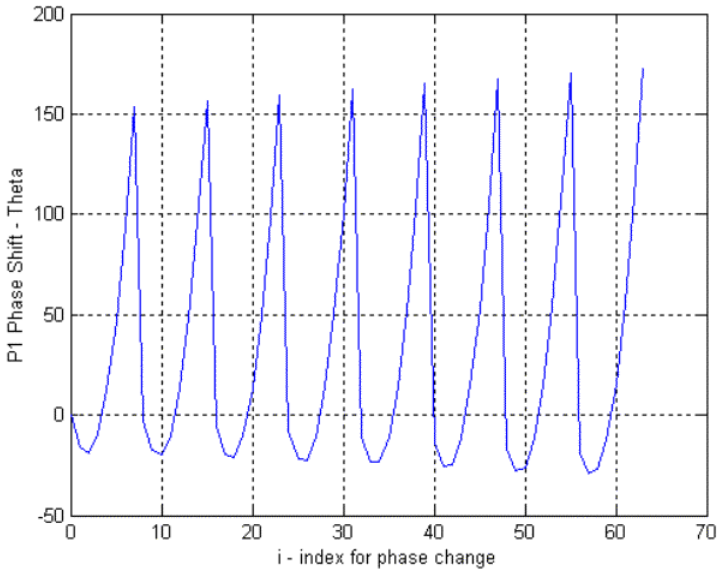


Figure 19 Phase shift for a P1-coded signal with N=8.

The PSD for a P1-coded signal is shown in Figure 20. Carrier frequency of 1000 Hz, and bandwidth of 1000 Hz can be identified in this figure. Figure 21 present the time domain description of the signal where the changes in phase can be observed. Figure 22 presents the PAF for a P1-coded signal. Figure 23 shows the cuts along the 0 Doppler and the 0 delay.

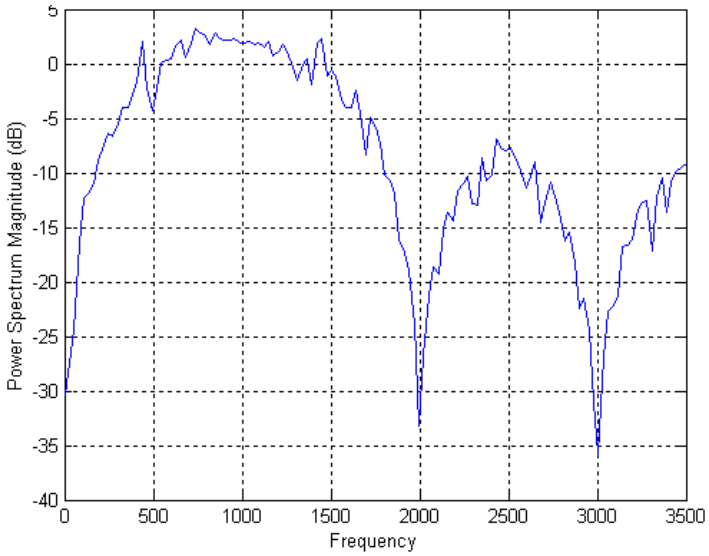


Figure 20 PSD for a P1-coded signal with *N*=8.

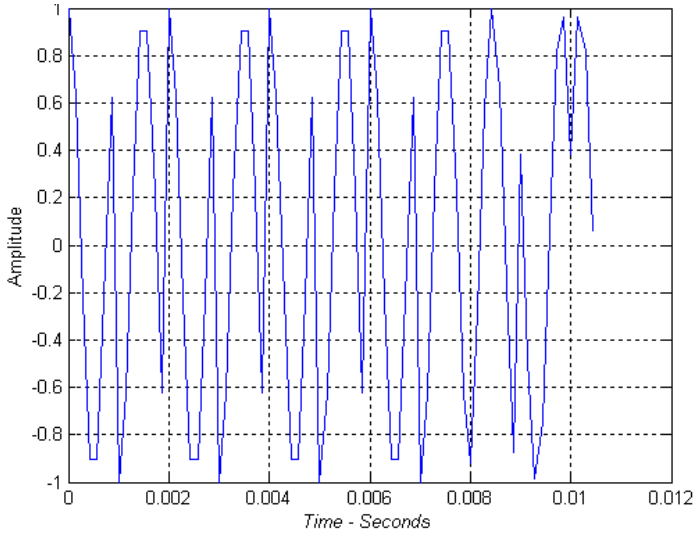


Figure 21 Time domain plot of a P1-coded signal with *N*=8.

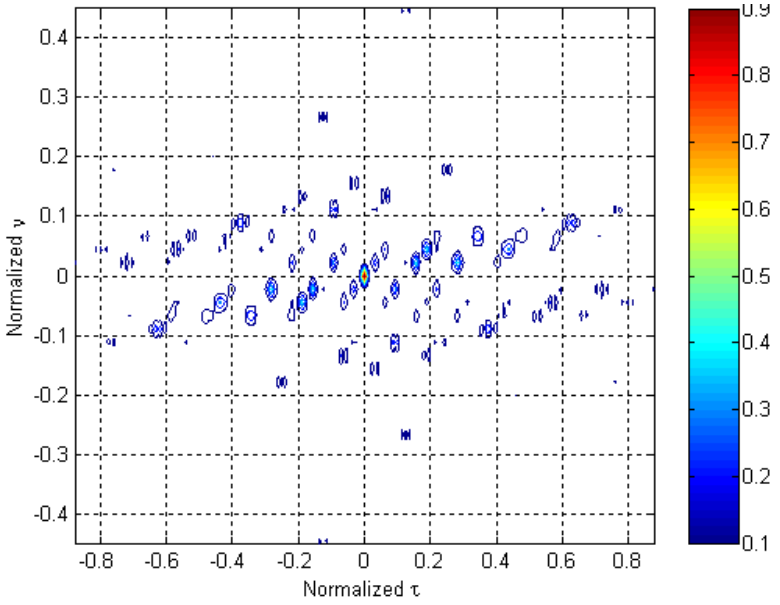


Figure 22 Contour plot of the Ambiguity Function for a P1-coded signal with *N*=8.

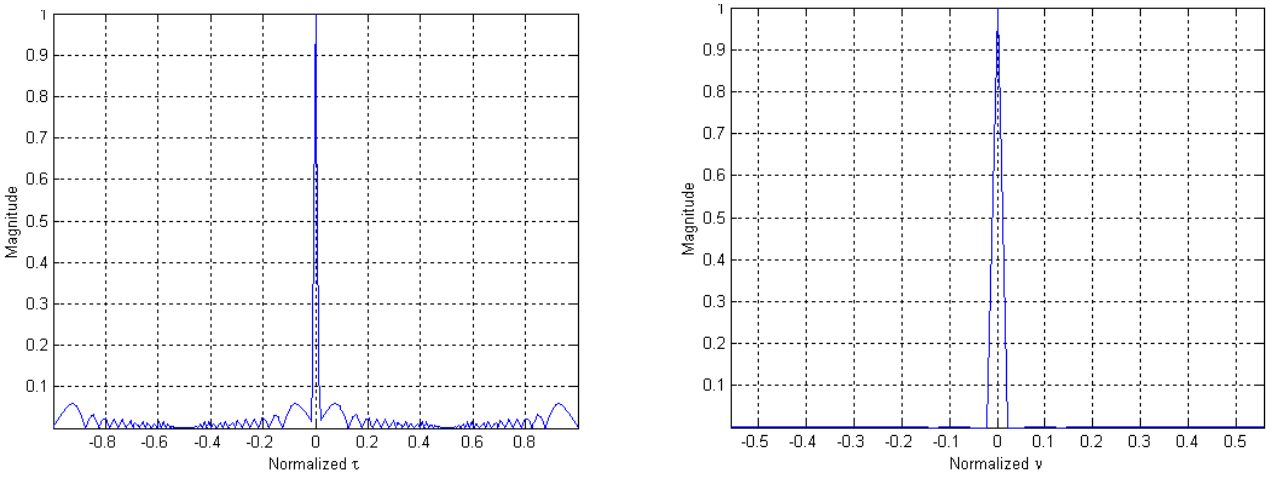


Figure 23 Cuts along the 0 Doppler and 0 delay axis of the PAF for a P1-coded signal with *N*=8.

E. P2 CODE

This code is essentially derived in the same way as the P1 code is derived. The P2 code has the same phase increments within each group as the P1 code, except that the starting phase is different. The P2 code is valid for N even, and each group of the code is symmetric about 0 phase. These phases can be calculated by

$$\phi_{i,j} = \left\{\frac{\pi}{2}[(N-1)/N] - (\frac{\pi}{N})(i-1)]\right\}[N+1-2j] \tag{2.1.9}$$

or

$$\phi_{i,j} = \frac{-\pi}{2N}[2j-1-N][2i-1-N] \tag{2.1.10}$$

where i=1, 2, …, N and j= 1, 2, …, N. This code has the frequency symmetry of the P1 code while also containing the property of being a palindromic code since the phases are symmetric in the center of the code. [6]

The P2 polyphase code, has more of a symmetrical frequency spectrum than a Frank-coded signal due to its symmetry in the carrier. Figure 24 shows the phase shift of a P2-coded signal with N=8 (N^2= 64 phases).

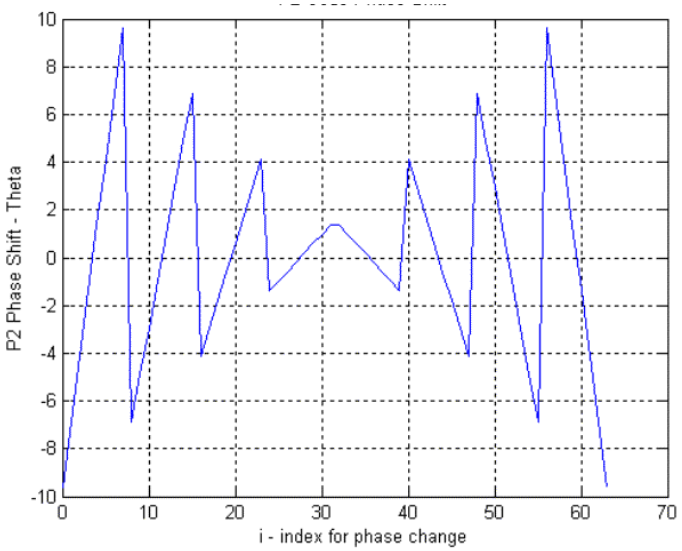


Figure 24 Phase shift for a P2-coded signal with N=8.

Figure 25 shows the PSD of a P2-coded signal with carrier frequency equal to 1000 Hz, 1 cycle per phase, bandwidth of 1000 Hz and N=8 (N^2 = 64 phases). Figure 26 presents a time domain representation of the signal where the phase shift can be observed.

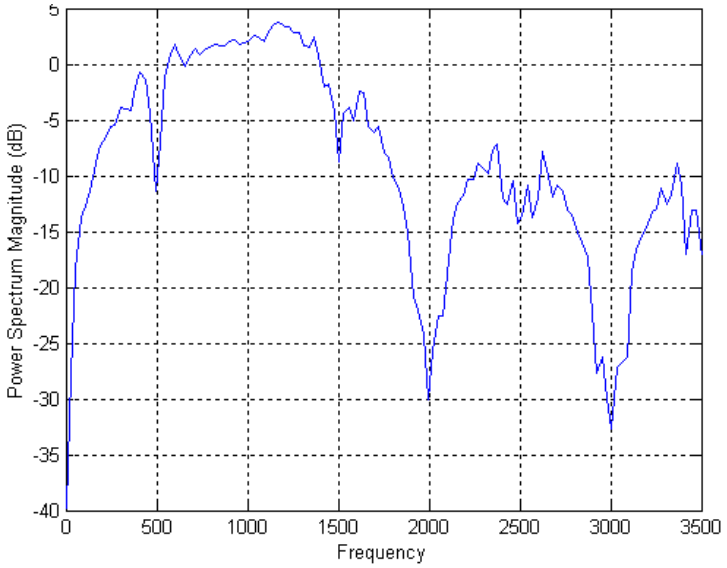


Figure 25 PSD for a P2 coded-signal with N=8.

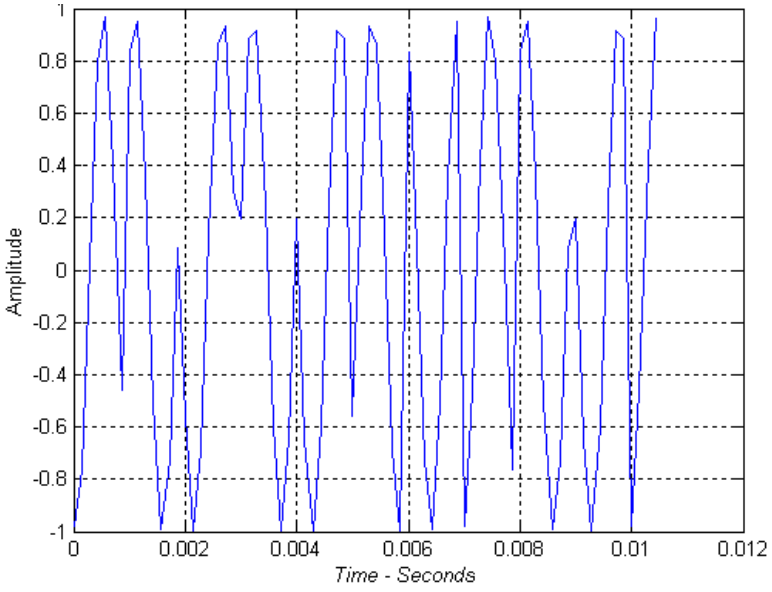


Figure 26 Time domain plot of a P2-coded signal with N=8.

The PAF of a P2-coded signal with *N*=8 is presented in Figure 27 . Additionally, Figure 28 shows the cuts along the 0 Doppler and the 0 delay axis. The PAF maximum side lobe levels of Frank, P1 and P2 are identical.

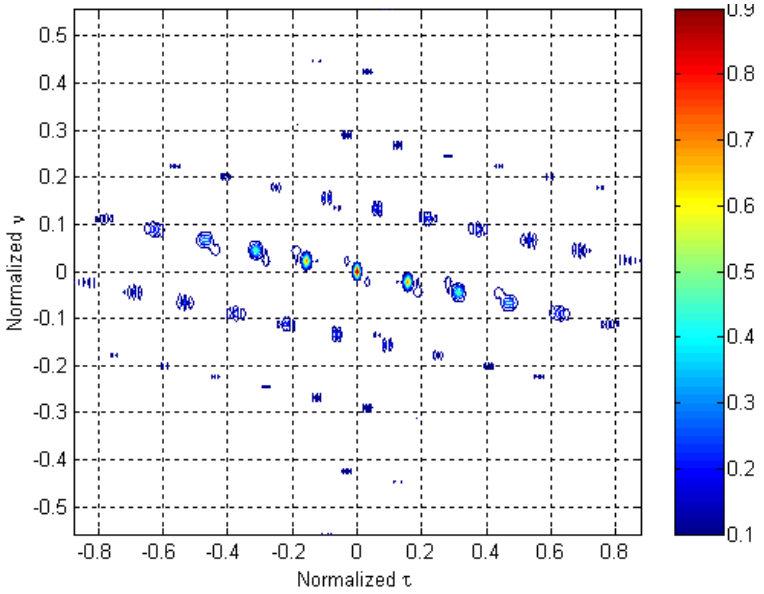


Figure 27 Contour plot of the PAF for a P2-coded signal with N=8.

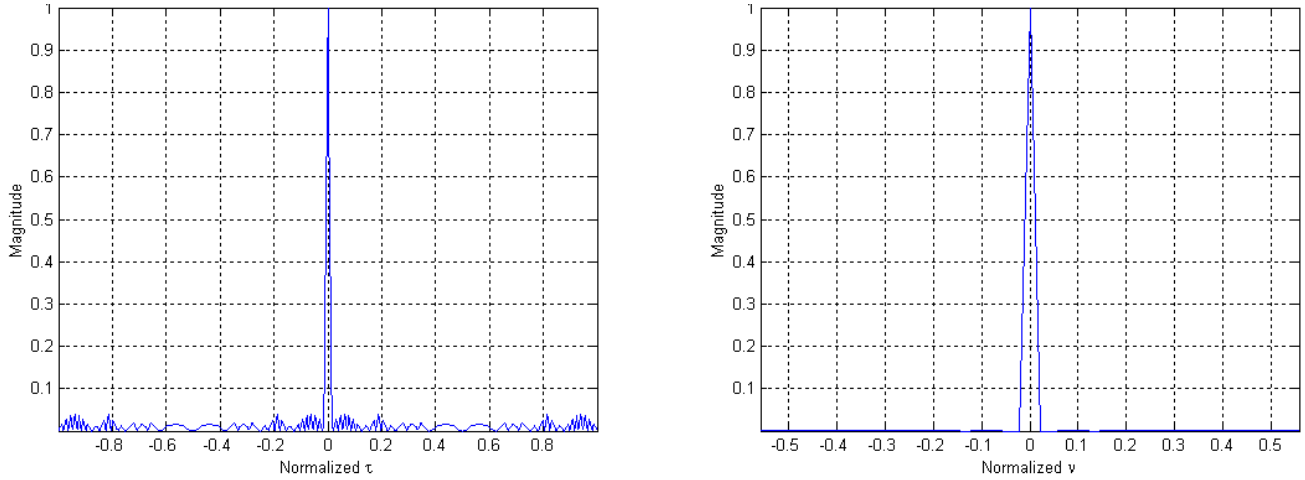


Figure 28 Cuts along the 0 Doppler and 0 delay axis of the PAF for a P2-coded signal with *N*=8.

F. P3 CODE

This code is derived by converting a linear-frequency modulation waveform to base band using a local oscillator on one end of the frequency sweep and sampling the *I* and *Q* video at the Nyquist rate. If it is assumed that the waveform has a pulse length *T* in frequency $f = f_o + kt$, where *k* is a constant, the bandwidth *B* of the signal will be approximately *B*=*kT*. [6]

The bandwidth will support a compressed pulse length of about $t_c = 1/B$ and the waveform will provide a pulse compression ratio of $pc = T/t_c = BT$. Assuming that the first sample of *I* and *Q* are taken at the leading edge of the waveform, the phases of successive samples taking t_c apart are

$$\phi_i = 2\pi \int_0^{(i-1)t_c} [(f_0 + kt) - f_0]dt \tag{2.1.11}$$

or

$$\phi_i = \pi k (i-1)^2 t_c^2 \tag{2.1.12}$$

if $k = \frac{B}{T}$, $i = 1, 2, ..., N$ and $t_c = \frac{1}{B}$, the equation can be written as

$$\phi_i = \frac{\pi (i-1)^2}{BT} = \frac{\pi (i-1)^2}{N} \tag{2.1.13}$$

Figure 29 shows the phase shift of a P3-coded signal with *N*=64.

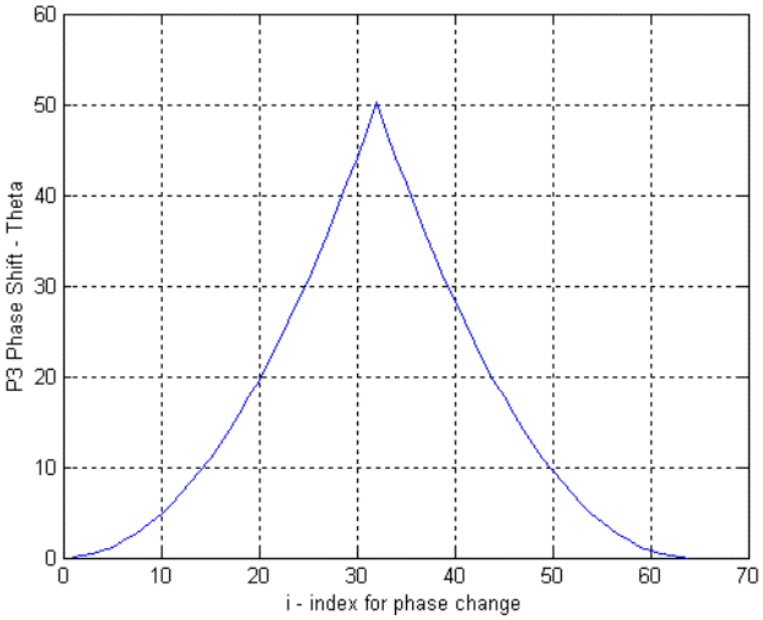


Figure 29 Phase shift for a P3-coded signal with N=64.

Figure 30 illustrates the PSD of the signal with carrier frequency of 1 KHz, 1 cycle per phase and bandwidth of 1 KHz. Figure 31 shows a time domain plot of the signal.

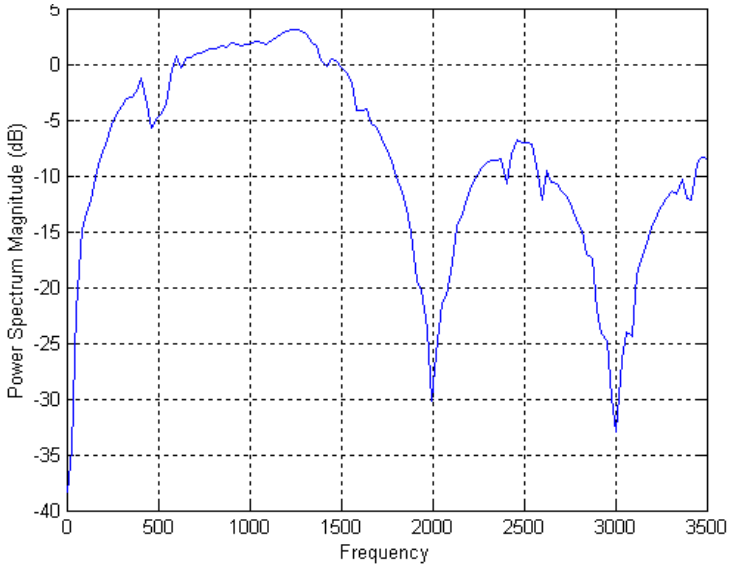


Figure 30 PSD for a P3-coded signal with N=64.

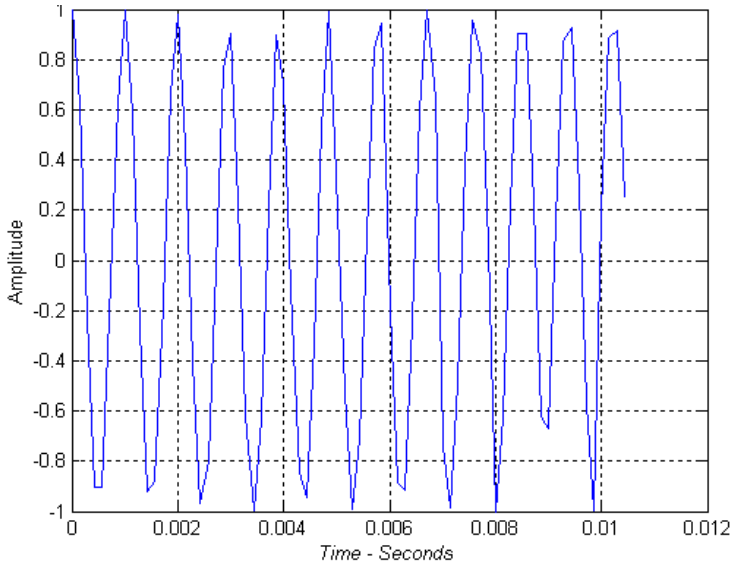


Figure 31 Time domain plot of a P3-coded signal with N=64.

The PAF of a P3-coded signal is shown in Figure 32 . Figure 33 presents the cuts along the 0 Doppler and 0 delay axis. The PAF of a P3 code is similar to that for Frank and P1 except that the peak sidelobes are approximately 4 dB higher.

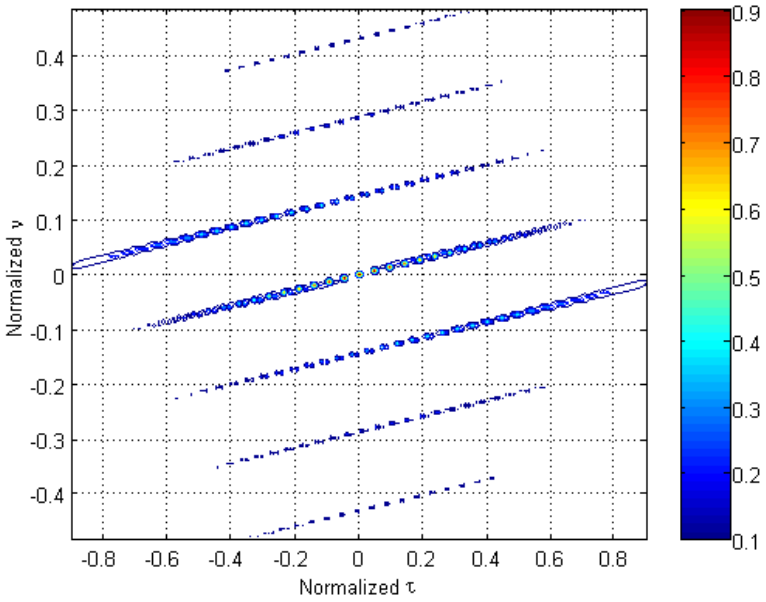


Figure 32 Contour plot of the PAF for a P3-coded signal with N=64.

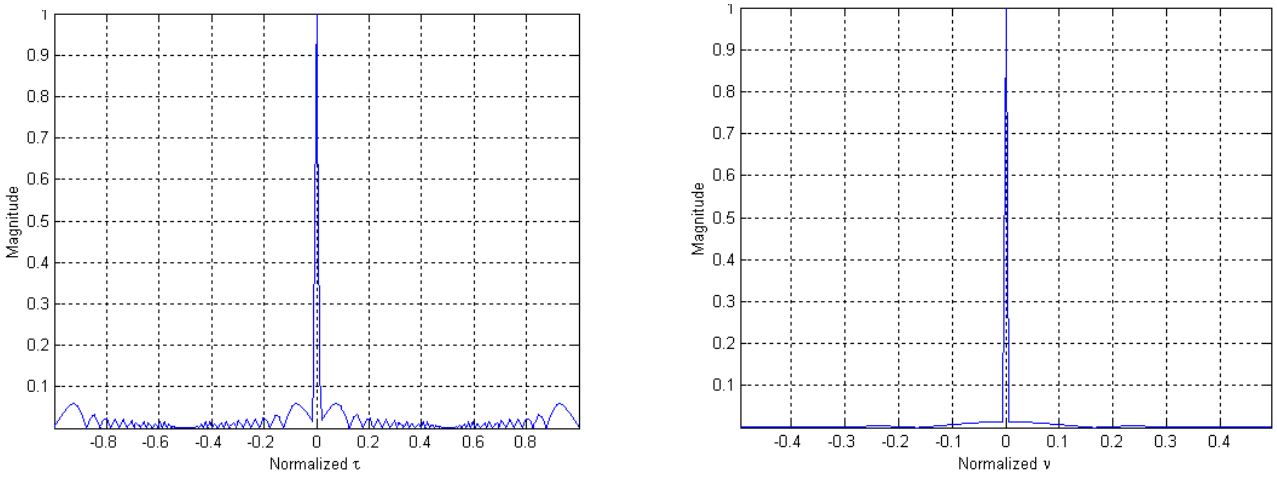


Figure 33 Cuts along the 0 Doppler and 0 delay axis of the PAF for a P3-coded signal.

G. P4 CODE

The P4 code is conceptually derived from the same waveform as the P3 code. However, in this case, the local oscillator frequency is set equal to $f_o + kT/2$ in the *I-Q* detectors [6] . With this frequency, the phases of successive samples taking t_c apart are

$$\phi_i = 2\pi \int_0^{(i-1)t_c} [(f_0 + kt) - f_0 + \frac{kT}{2}]dt \qquad (2.1.14)$$

or

$$\phi_i = 2\pi \int_0^{(i-1)t_c} k(t - T/2)dt \qquad (2.1.15)$$

or

$$\phi_i = \pi k(i-1)^2 t_c^2 - \pi kT(i-1)t_c = \left[\frac{\pi(i-1)^2}{N}\right] - \pi(i-1). \qquad (2.1.16)$$

Figure 34 shows the relationship between the index in the matrix and its subsequent phase for a P4-coded signal with N= 64 (phases).

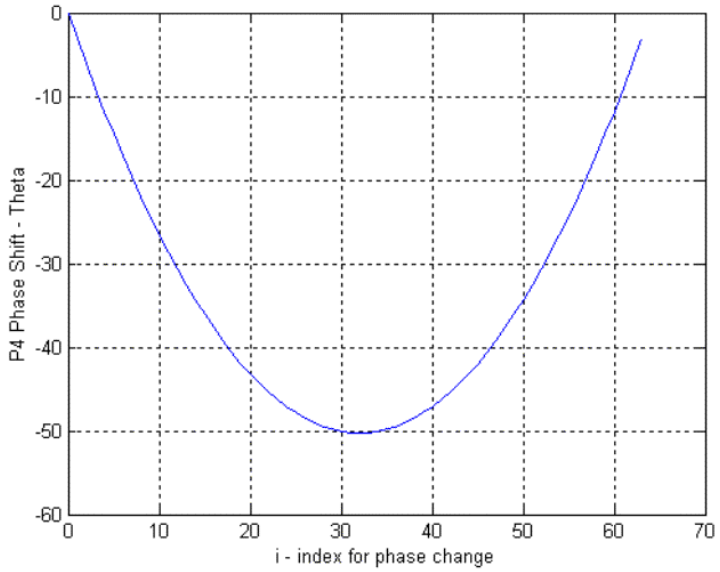


Figure 34 Phase shift for a P4-coded signal with N=64 phases.

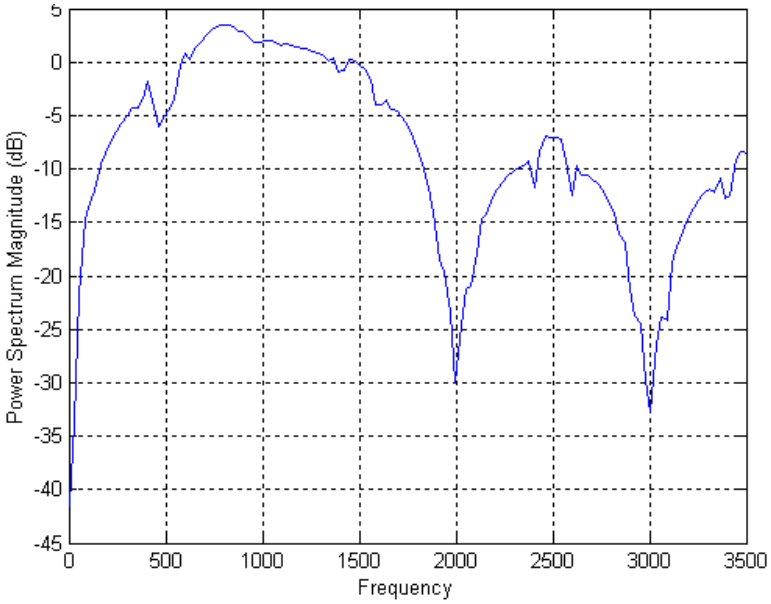


Figure 35 PSD for a P4-coded signal with N=64.

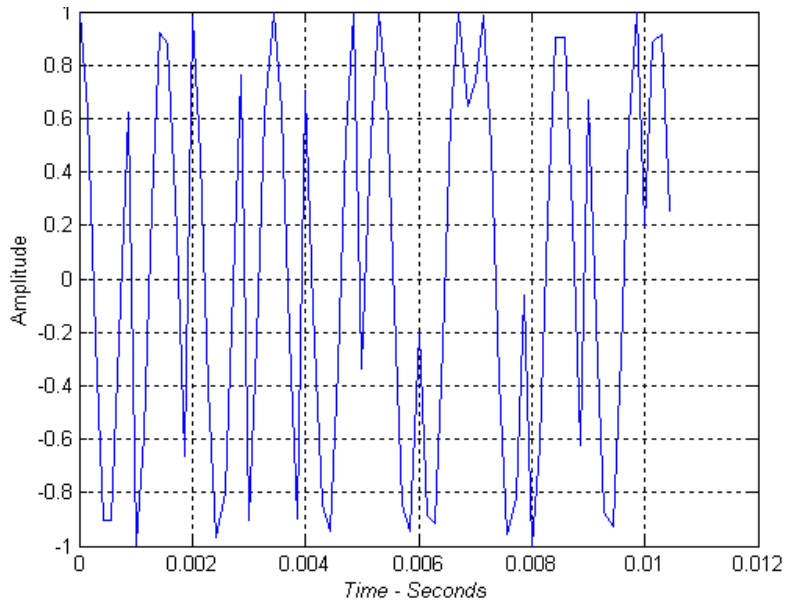


Figure 36 Time domain plot of a P4-coded signal with N=64.

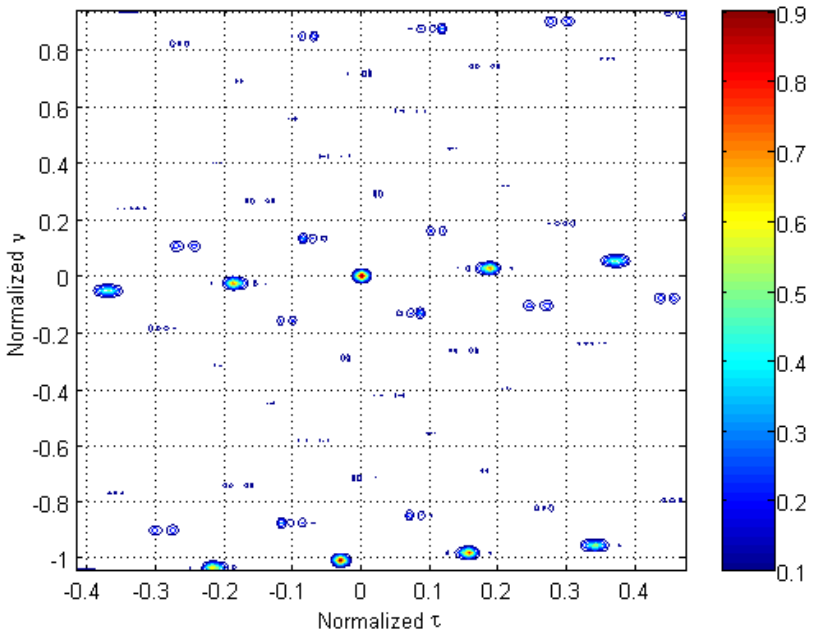
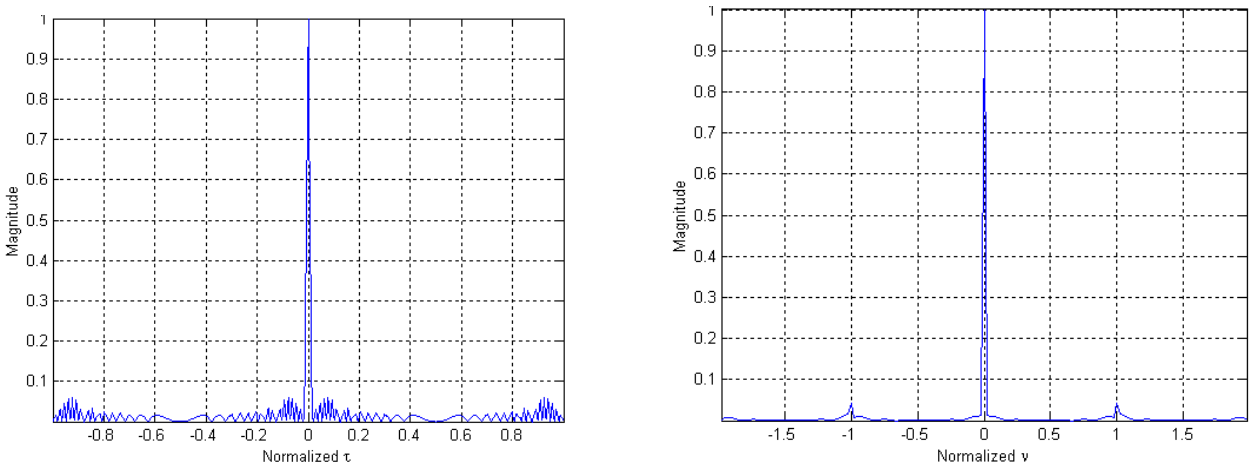


Figure 37 Contour plot of the PAF for a P4-coded signal with N=64.



(a) (b)

Figure 38 P4-coded signal PAF (a) Cuts along the 0 Doppler and (b) 0 delay axis.

H. COSTAS CODE

In a frequency hopping system, the signal consists of one or more frequencies being chosen from a set $(f_1, f_2, ..., f_m)$ of available frequencies, for transmission at each of a set $(t_1, t_2, ..., t_n)$ of consecutive time intervals. For modeling purposes, it is reasonable to consider the situation in which $m=n$, and a different one of n equally spaced frequencies $(f_1, f_2, ..., f_m)$ is transmitted during each of the n equal duration time intervals $(t_1, t_2, ..., t_n)$. Such a signal is represented by a n x n permutation matrix A, where the n rows correspond to the n frequencies, the n columns correspond to the n intervals, and the entry $a_{i,j}$ equals 1 means transmission and 0 otherwise[7] . This signifies that, at any given time, a tone frequency is transmitted, and each frequency only once as shown in Figure 39(a). Other possible frequency-hopping sequences belong to this family. This hopping order stongly affects the ambiguity function of these signals. Costas frequency-hopping signals allow a simple procedure that results in a rough approximation of their ambiguity function.

Frequency

0	0	0	0	0	**1**
0	0	0	0	**1**	0
0	0	0	**1**	0	0
0	0	**1**	0	0	0
0	**1**	0	0	0	0
1	0	0	0	0	0

Time
(a)

Frequency

0	0	0	**1**	0	0
0	**1**	0	0	0	0
0	0	0	0	**1**	0
0	0	0	0	0	**1**
1	0	0	0	0	0
0	0	**1**	0	0	0

Time
(b)

Figure 39 Binary matrix representation of (a) quantized linear FM and (b) Costas Signal

This is possible because the cross correlation signals at different frequencies approach zero when the frequence difference is large relative to the inverse of the signal duration. The PAF, at any given coordinates, is an integral of the product between the original signal and a replica of it, which is shifted in time and frequency according to the delay and the Doppler coordinates of the function.

From the results of the difference matrix in Figure 39(b) except for the zero-shift cases, when the number of coincidences is *N*, finding a combination of shifts yielding more than one coincidence is not possible. This is actually the criteria of Costas sequences, where sequences of frequency-hopping yield no more than one coincidence. For example if $\{a_j\} = 2,6,3,8,7,5,1$ is a Costas matrix, then its coding matrix and difference matrix are shown in Figure 40 .

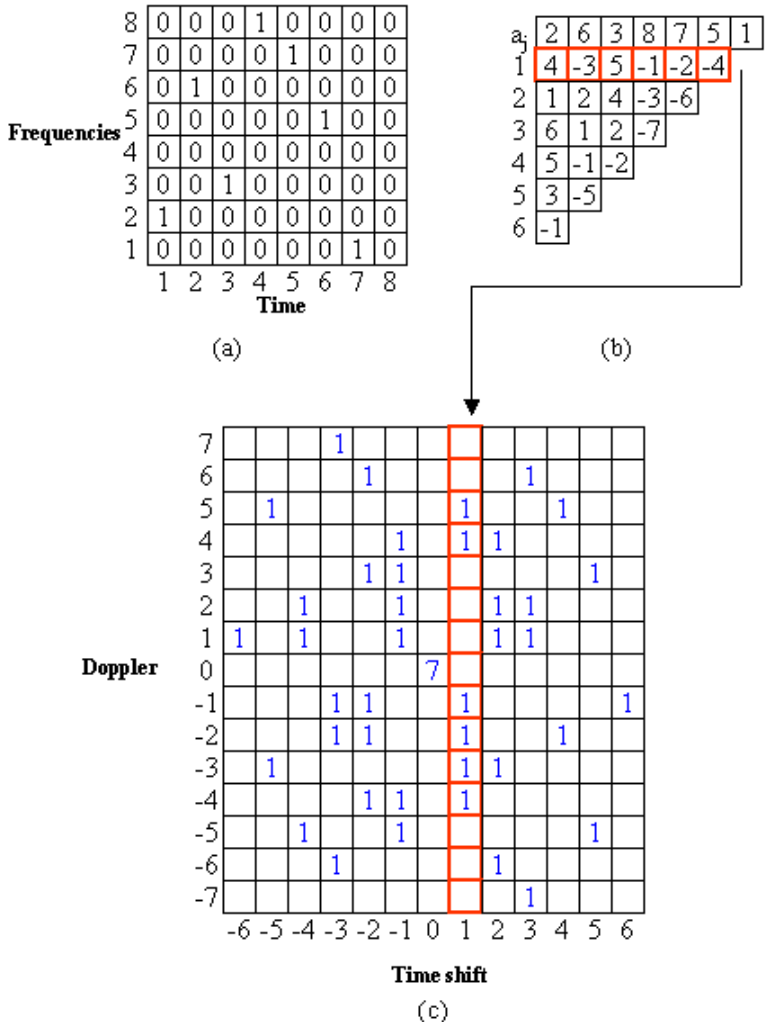


Figure 40 (a) The coding matrix (b) difference matrix and (c) ambiguity sidelobe matrix of a Costas signal.

Figure 41 illustrates the PSD of the Costas-coded signal. Seven frequencies are observed in the power distribution of the signal along the frequencies. Figure 42 shows a portion of the time domain representation of the coded signal.

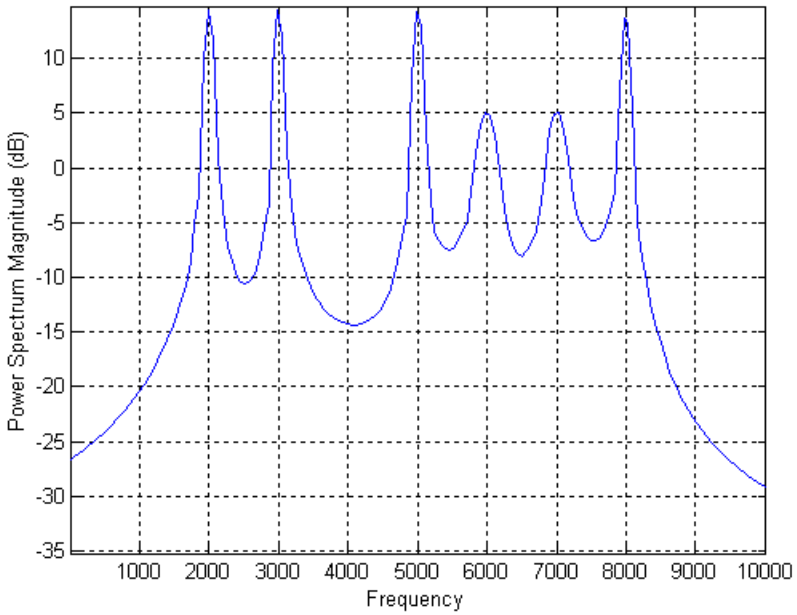


Figure 41 PSD for a Costas signal.

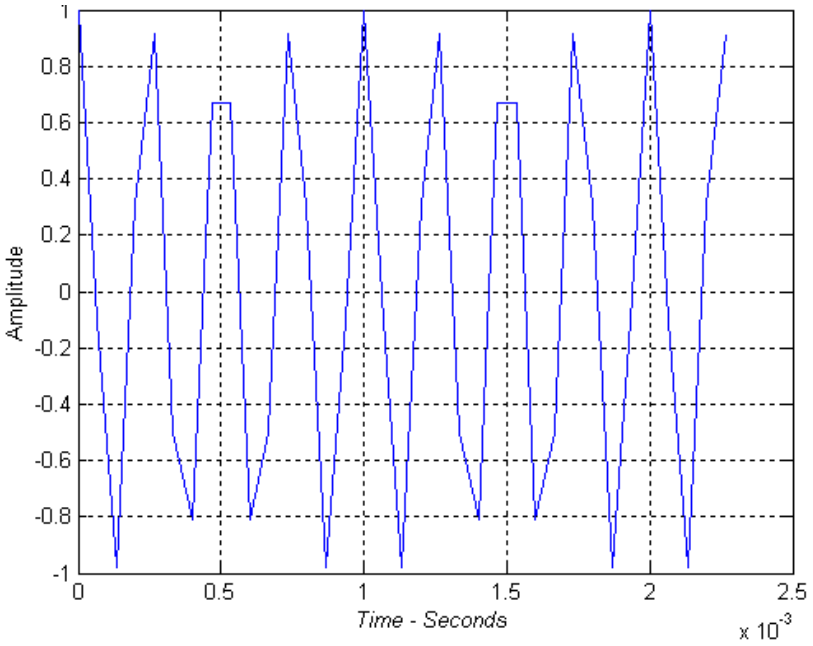


Figure 42 Time domain plot of a Costas-coded signal.

Figure 43 through Figure 46 present the PAF for a Costas-coded signal where the distribution of the sidelobes can be compared with the sidelobes matrix in Figure 40 .

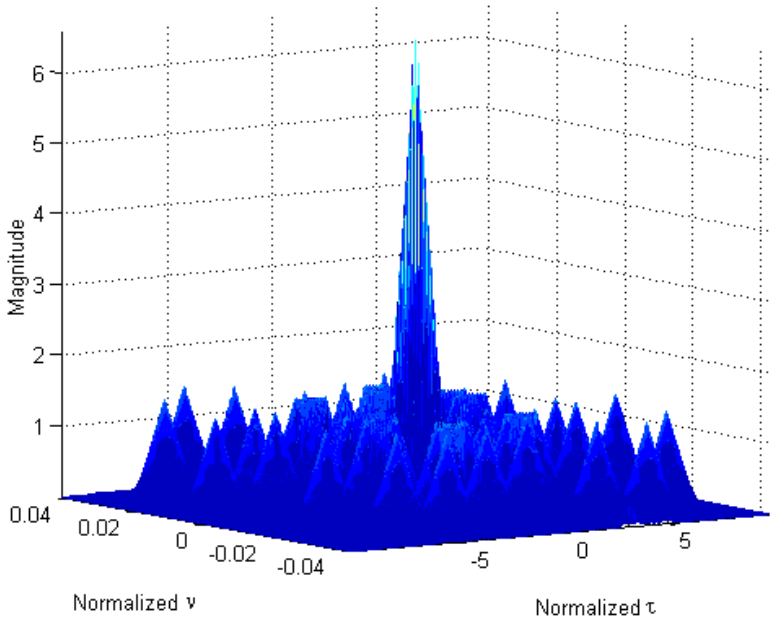


Figure 43 3-D PAF for a Costas-coded signal with sequence 2-6-3-8-7-5-1.

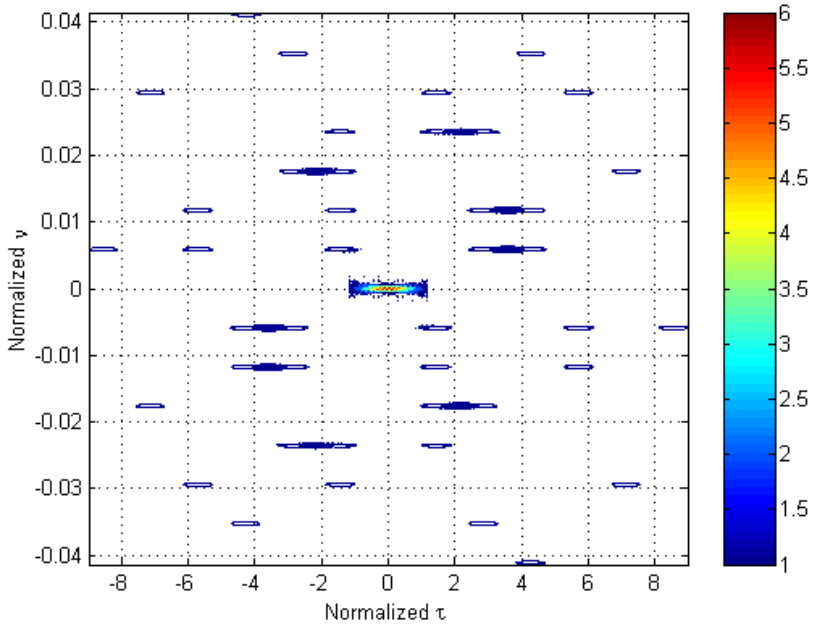


Figure 44 Contour plot of the PAF for the Costas-coded signal with sequence 2-6-3-8-7-5-1.

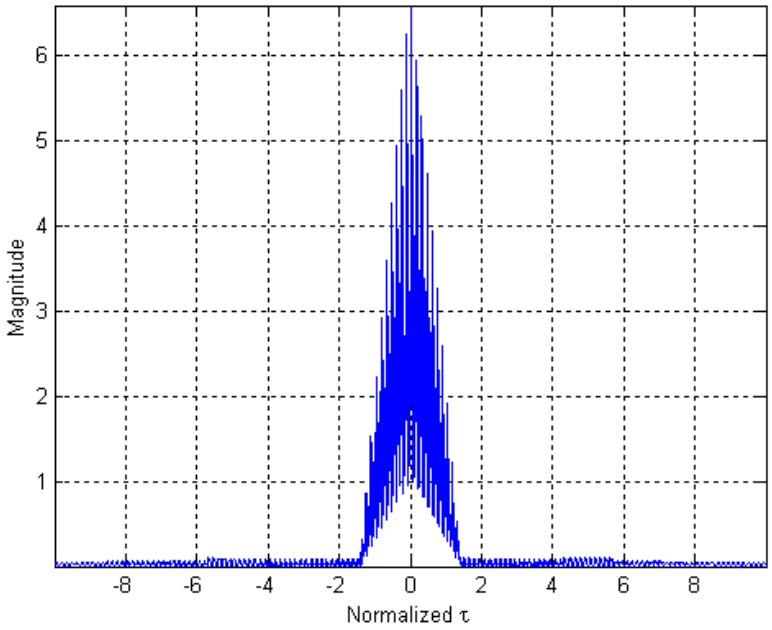


Figure 45 Cut along the 0 Doppler axis.

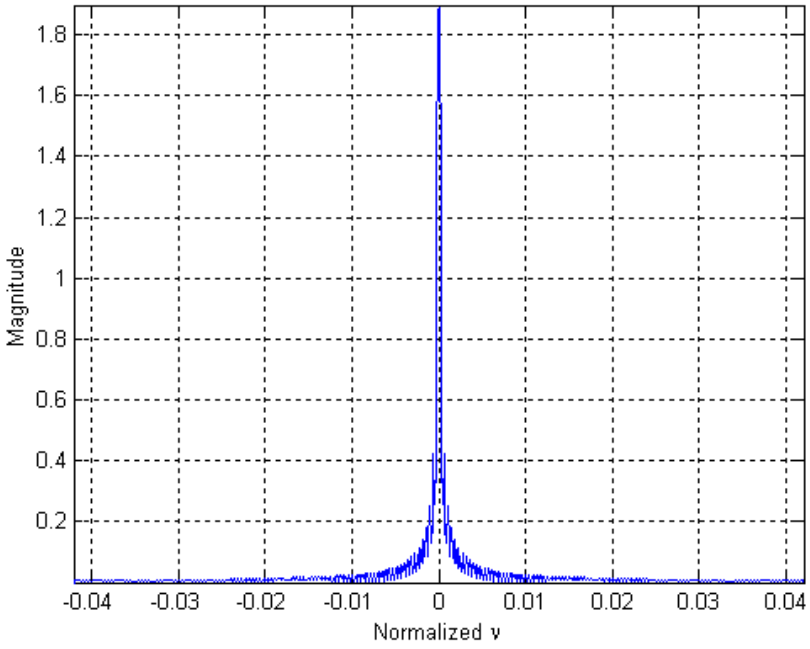


Figure 46 Cut along the 0 delay axis.

I. PHASE SHIFT KEYING/FREQUENCY SHIFT KEYING.

This modulation technique is the result of a combination of frequency shift keying based on a Costas frequency hopping matrix and phase shift keying using Barker sequences of different lengths. In a Costas frequency-hopped signal, the firing order of the N_F frequencies each with sub-period T_F defines what frequencies will appear and with what duration. During each sub-period, as the signal stays in one of the frequencies, a binary phase modulation occurs according to a Barker sequence of length N_P = 5, 7, 11 or 13. The final waveform may be seen as a binary phase shift modulation within each frequency hop.[8]

As illustrated in Figure 1, if we consider N_F frequency hops and N_P as the number of phase slots of duration T_P in each sub-period T_F, the total number of phase slots in the FSK/PSK waveform is given by [16]

$$N = N_F N_P \tag{2.1.17}$$

The block diagram in Figure 47 (a), describes the MATLAB® implementation. The user defines which sequence of Costas frequency hops he wants to use and also how long the Barker sequence is (5, 7, 11 or 13). The number of frequency hops is pre-defined to be seven and the user may select from seven different options of sequences of frequencies varying from 1 KHz to 11 KHz. The Barker sequence is generated and the frequency hopping signal is then modulated accordingly.

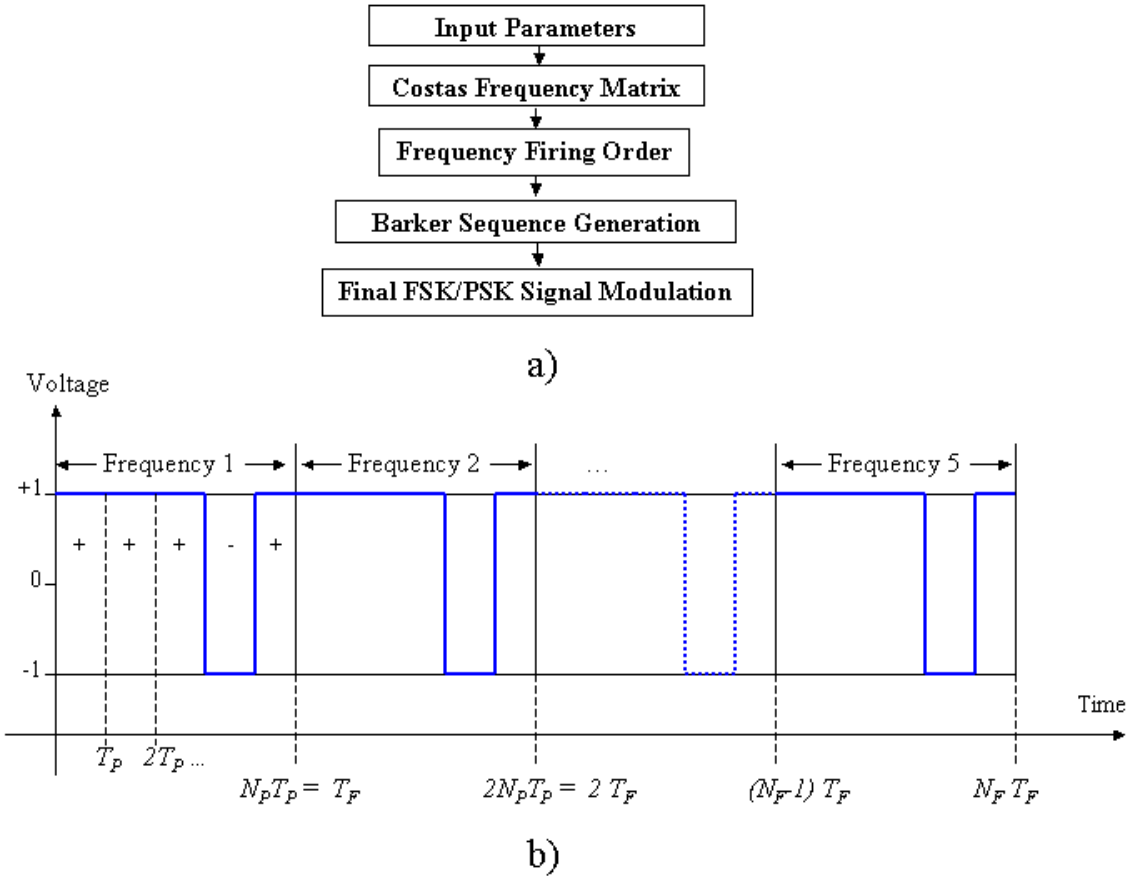


Figure 47 FSK/PSK (a) block diagram, (b) General FSK/PSK signal containing N_F frequency hops with N_P phase slots per frequency. (from [8]).

Figure 48 shows the Costas frequency-hopping waveform PSD before it is phase modulated. Figure 49 presents the PSD for a FSK/PSK Costas-coded signal after phase modulation. The PSD plots reveal the spread spectrum characteristic of these signals. The Costas sequence is always seven frequency hops (4, 7, 1, 6, 5, 2, and 3 KHz). The sampling frequency is 15 KHz, satisfying the Nyquist minimum rate ($f_s >> 2f_c$) for the biggest frequency value. All plots are generated in MATLAB® using the routines **fsk_psk_costas.m** and **PAF_FSK_PSK.m**, both listed in Appendix A.

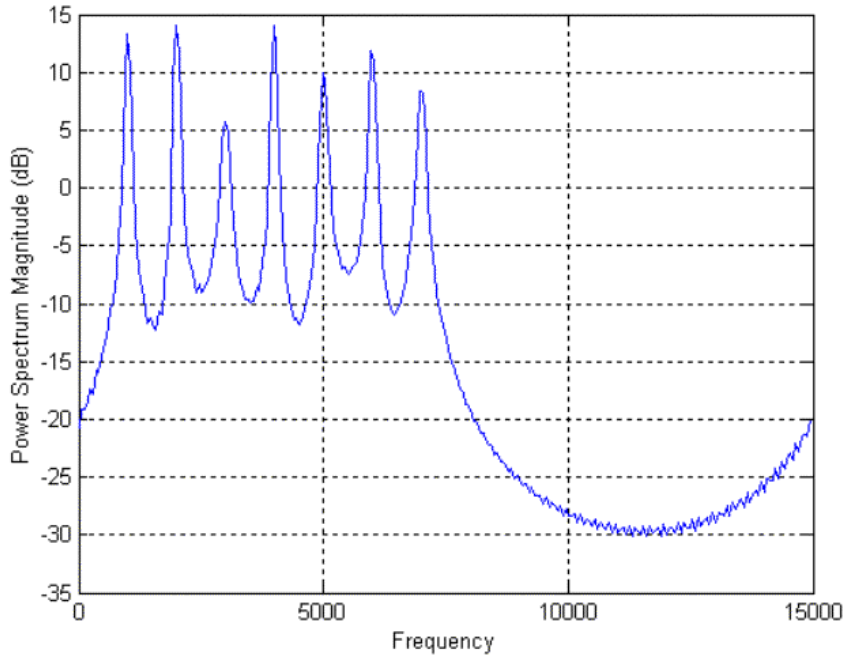


Figure 48 PSD for a Costas-coded signal.

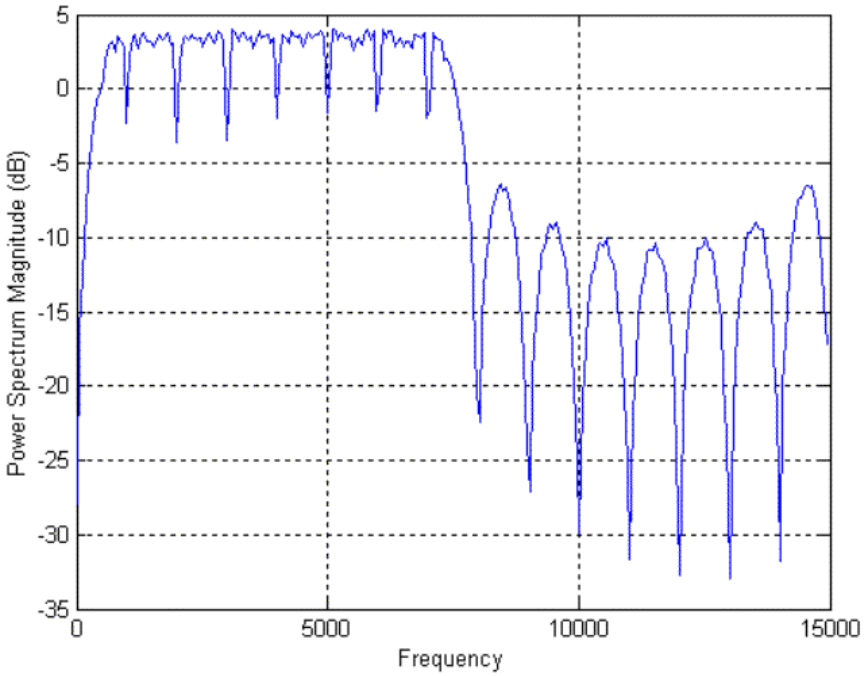


Figure 49 PSD of a FSK/PSK Costas-coded signal.

J. FSK/PSK COMBINED WITH TARGET-MATCHED FREQUENCY HOPPING

Instead of spreading the energy of the signal equally over a broad bandwidth, the target matched frequency hopping technique concentrates the signal energy in specific spectral locations that are important for the radar target detection capability within its broad-spectrum bandwidth. The transmitted signals have a pulse compression characteristic and therefore they can achieve low probability of intercept.

Figure 50 illustrates the block diagram for the generation of FSK/PSK in addition, Figure 51 shows the FSK/PSK target simulated response and the probability distribution and frequency firing order with the number of occurrences per frequency. The implementation starts with a simulated target time radar response. This data is then Fourier transformed and the correspondent N_F frequencies and initial phases are collected. A random selection process chooses each frequency with a probability distribution function defined by the spectral characteristics of the target of interest (obtained from the FFT). The frequencies with the highest spectral peaks (largest magnitudes) are transmitted more often. Each 'frequency hop,' transmitted is also modulated in phase, having its initial phase value modified by a pseudo-random phase sequence of values equally likely to be zero or π radians. [8]

The matched FSK/PSK radar will then use a correlation receiver with a phase mismatched reference signal instead of a perfectly phase matched reference. This allows the radar to generate signals that can match a target's spectral response in both magnitude and phase.

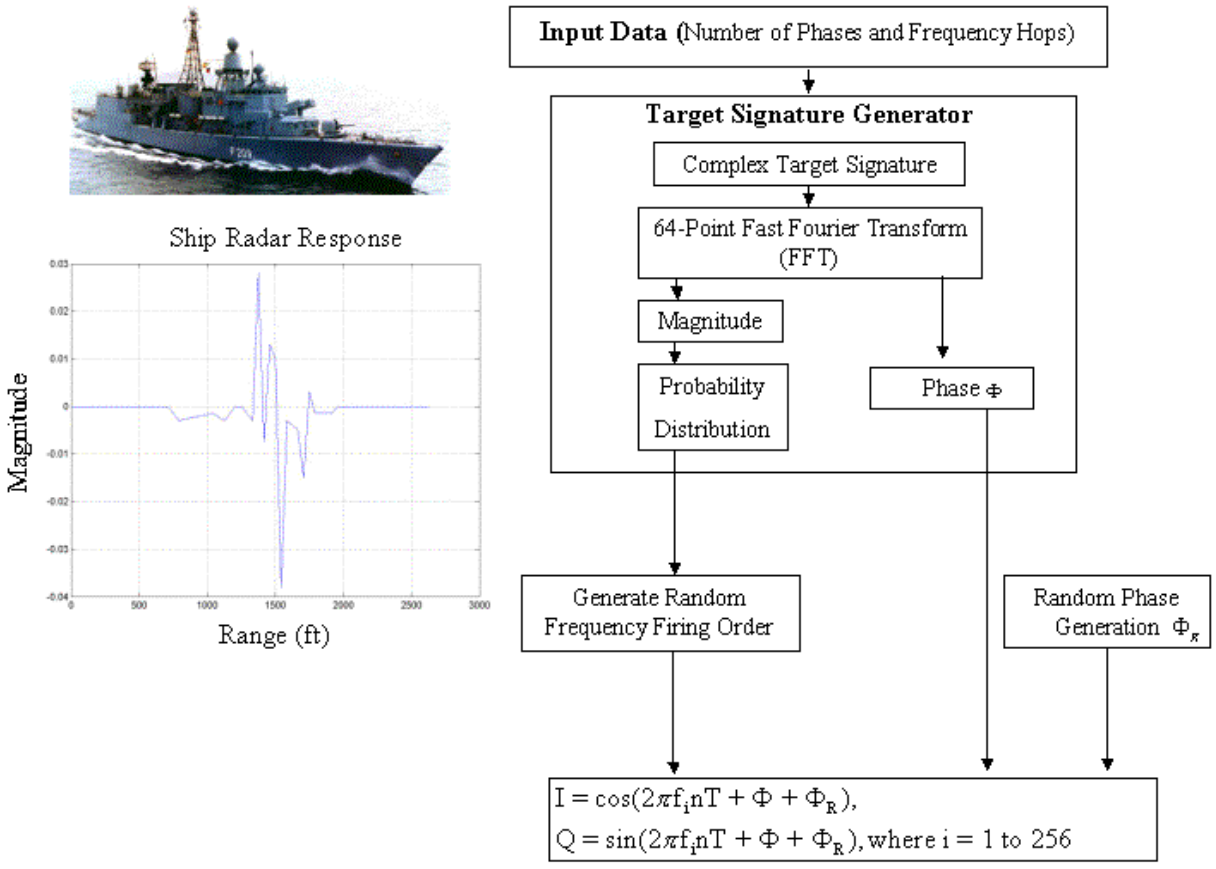
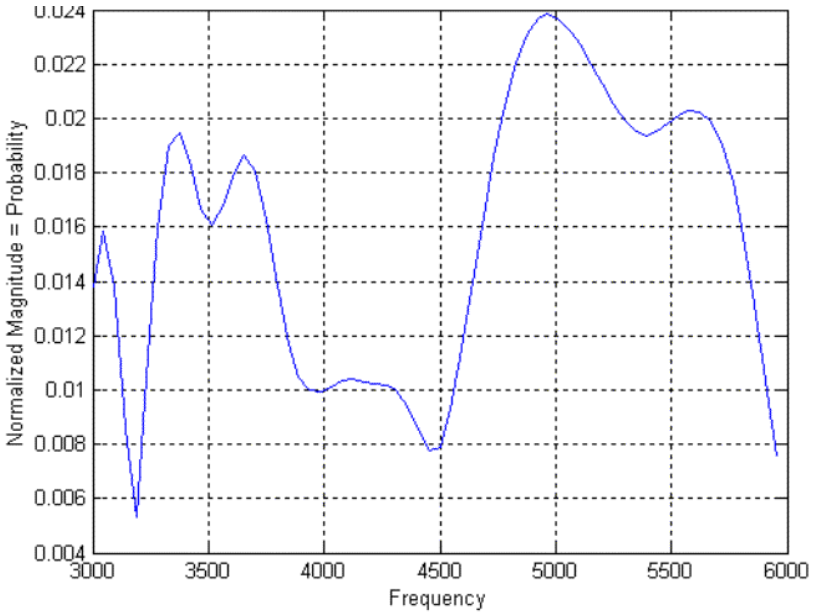
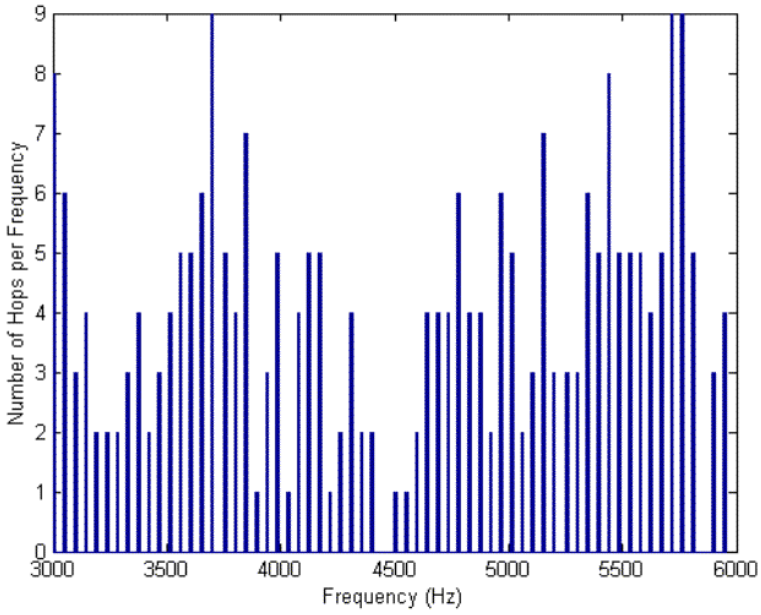


Figure 50 Block diagram of the implementation of the FSK/PSK Target matched waveform.



(a)



(b)

Figure 51 (a) FSK/PSK target 64 frequency components and frequency probability distribution (b) FSK/PSK target 64 frequency components histogram with number of occurrences per frequency for 256 frequency hops

K. LOW PROBABILTY OF INTERCEPT SIGNAL GENERATOR

A signal generator toolbox was developed due to the necessity of creating the input signals with LPI characteristics. The purpose of the LPI signal generator toolbox is to generate a variety of LPI signals and evaluate their time-domain and frequency-domain characteristics. This toolbox by itself is an important contribution for future research in this area. The program, implemented in MATLAB® 6.1, can generate the following LPI signals and test signals:

1. Binary Phase Shift Keying (BPSK)
2. Frequency Modulated Continuous Wave (FMCW)
3. Frank-coded signals
4. P1
5. P2
6. P3
7. P4
8. Costas-coded signal
9. Frequency Shift-Keying/Phase Shift-Keying (FSK/PSK) Costas
10. Frequency Shift-Keying/Phase Shift-Keying (FSK/PSK) Target
11. Test signal – single tone and two tones

The block diagram of the LPI signal generator is shown in Figure 53. The total LPI generator toolbox consists of one main program and eleven (11) sub-programs with important general features, such as the capacity to add white Gaussian noise. The noise is added to the signal by specifying a desired SNR as

$$SNR = \frac{A^2}{2\sigma^2} \tag{2.1.18}$$

where A is the amplitude and σ^2 is the WGN. The idea of developing modular programs can facilitate the inclusion of new modulations without complex changes in the rest of the sub-programs. The MATLAB® code is presented in Appendix A.

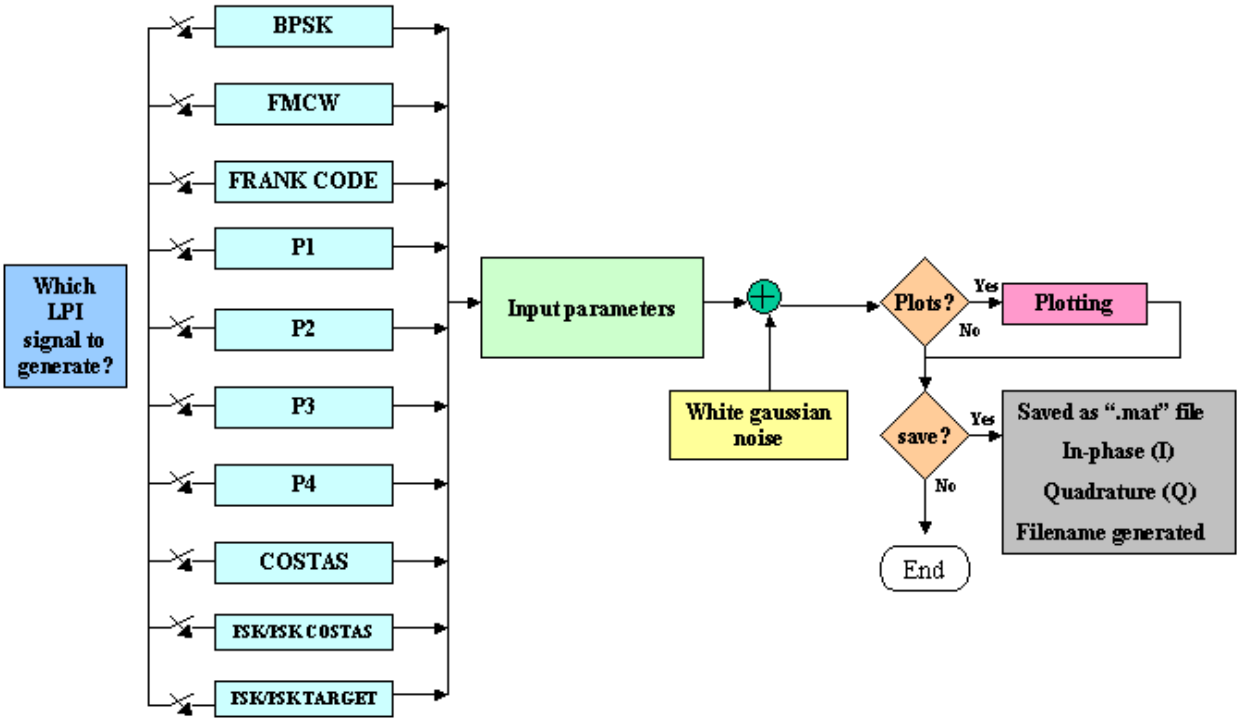


Figure 52 Block diagram of the LPI signal Generator.

1. LPI Signal Generator – Main Program

This program begins by creating a main menu (Figure 53) and includes all the available signals, so the user can generate signals continuously without opening each sub-program separately. Because of the modular design, the user can use each sub-program individually if desired. The name of the program is **LPIG.m**, which can be found in Appendix A.

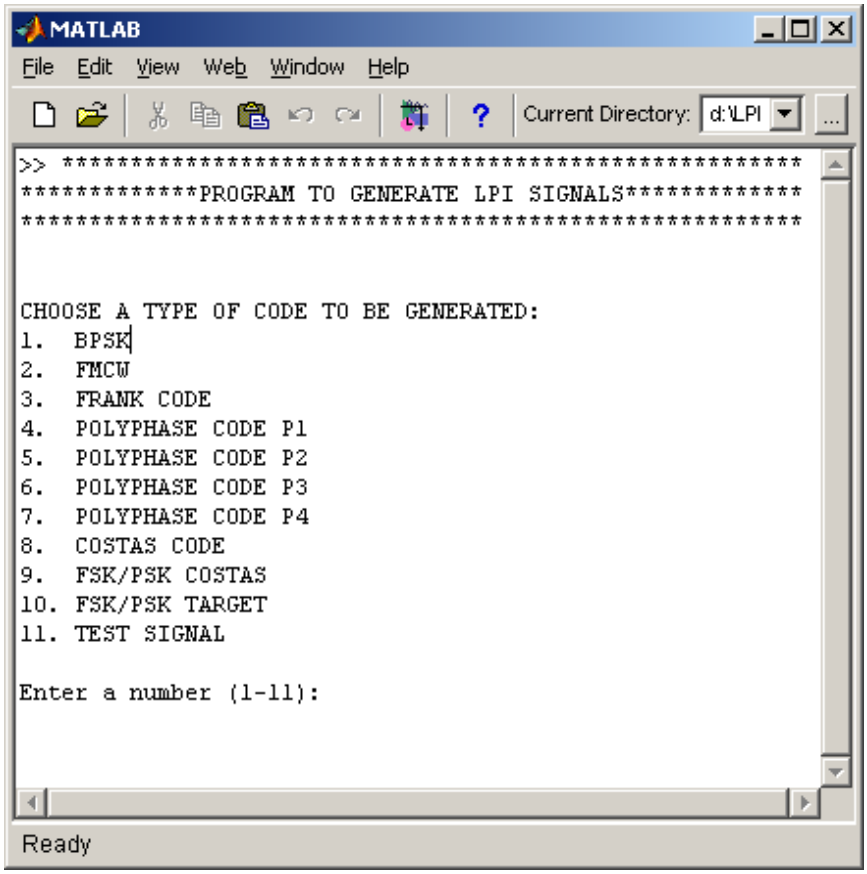


Figure 53 LPI signal generator main menu.

2. BPSK

The objective of this sub-program is to generate test signals with Binary Phase Shift-Keying determined by *n*-bit Barker code. This program, identified as **bpsk.m**, can be run from **LPIG.m** (main menu) or directly. The execution of the program guides the user to the BPSK sub-menu, where the following parameters can be adjusted (Figure 54):

Amplitude of the carrier signal
Carrier frequency (Hertz)
Sampling frequency (Hertz)
Signal to noise ratio (dB)
Number of bits per Barker code (7,11 or 13-bit)
Number of code periods. The code will be generated as often as requested.
Number of cycles per bit. Number of cycles of the carrier frequency for each bit of the Barker sequence.
Number of code periods to view on graph.

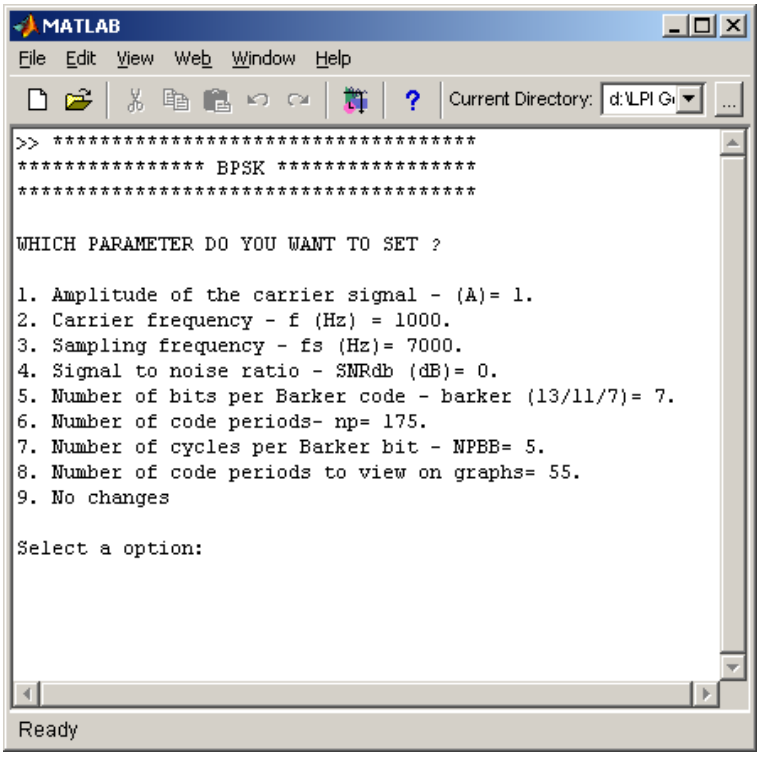


Figure 54 BPSK menu.

The BPSK signal generator produces different plots to show the main characteristics, facilitating understanding and analysis of the resulting signals. The results include the PSD of the Barker sequence, PSD of the modulated signal and time domain representation.

The output of the signal generator is saved with an automatically-generated name. The file contains the In-Phase and Quadrature components of the signal. In addition, the program provides the directory path where the signal was saved (the same directory as he program files).

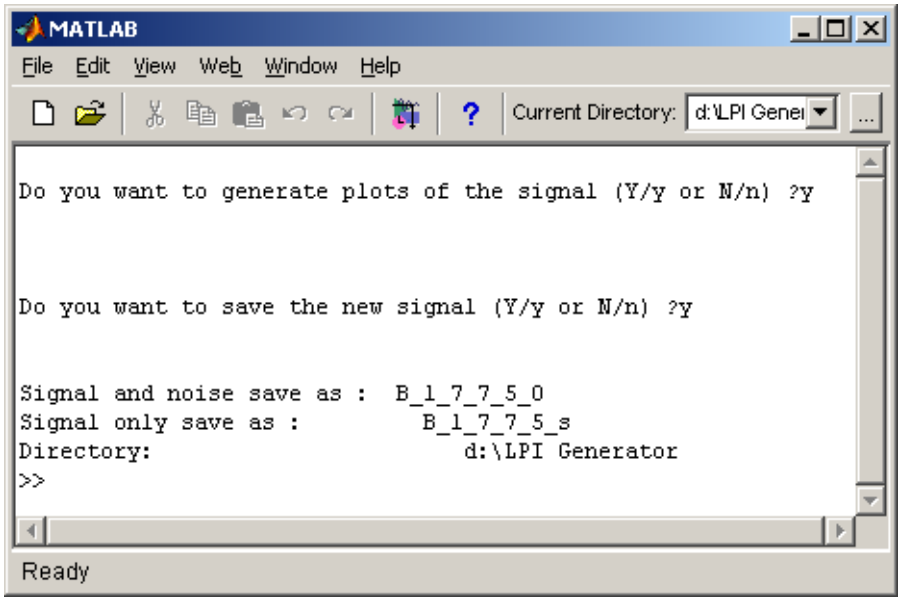


Figure 55 BPSK Signal Generator screen shot.

3. FMCW

The objective of this sub-program is to generate triangular FMCW signals. This program, named **fmcw.m**, can be run from **LPIG.m** (main menu) or run directly. The execution of the program guides the user to the FMCW sub-menu, where some parameters that can be adjusted as shown in Figure 56 :

1. Amplitude of the carrier signal
2. Carrier frequency (Hertz)
3. Sampling frequency (Hertz)
4. Signal to noise ratio (dB)
5. Modulation bandwidth (Hertz)
6. Modulation period (seconds)
7. Number of triangles to be generated.

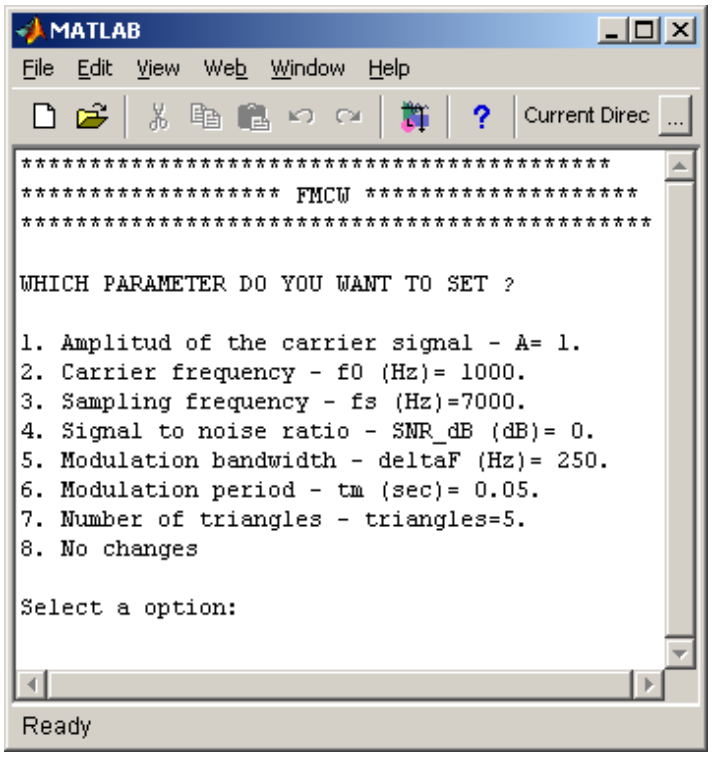


Figure 56 FMCW sub- menu.

The output of the signal generator can be saved with an automatically-generated name. The file contains the In-Phase and Quadrature components of the signal. In addition, the program provides the directory where the signal was saved as shown in Figure 58.

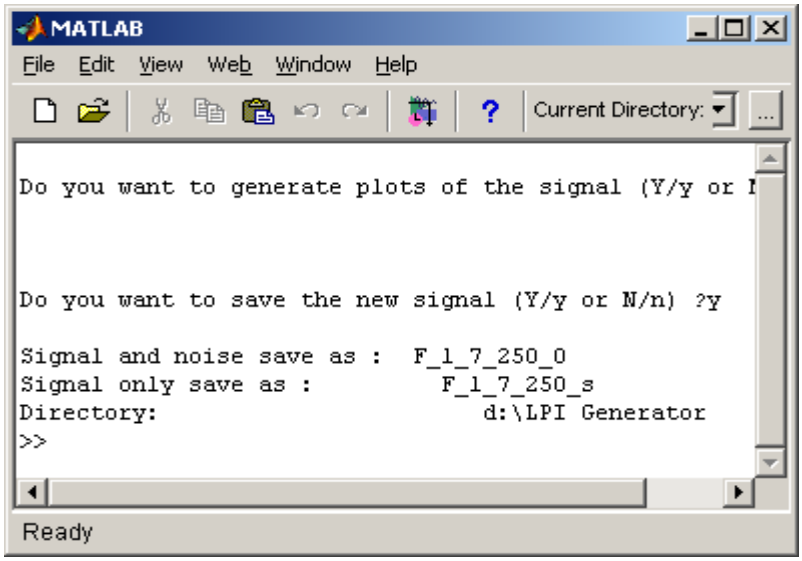


Figure 57 FMCW Signal generator screen shot.

4. Polyphase-coded signals: Frank code, P1, P2, P3 and P4

The objective of these programs is to generate LPI signals using polyphase coding. These programs named **frank.m, p1.m, p2.m, p3.m and p4.m** can be run from **LPIG.m** (main menu) or run directly. The execution of these programs leads the user to the corresponding sub-menus, where many parameters can be adjusted: (see Figure 58)

8. Amplitude of the carrier signal.
9. Carrier frequency (Hertz)
10. Sampling frequency (Hertz)
11. Signal to noise ratio (dB)
12. Number of phase codes
13. Number of cycles per phase.

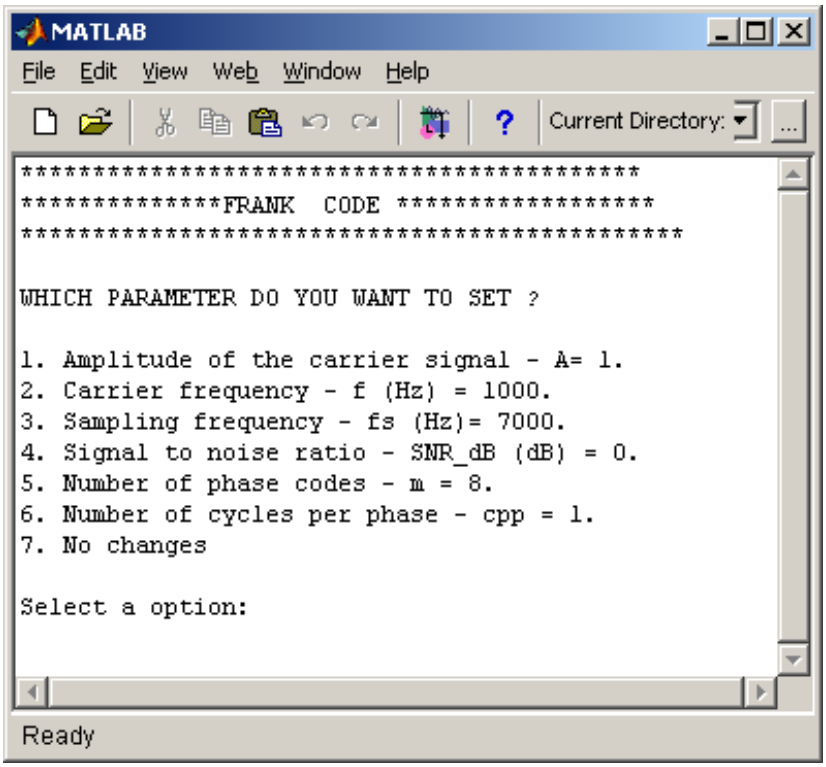


Figure 58 Frank-coded Signal Generator main menu. The sub-menu for generating all the polyphase signals are equal.

The polyphase-coded signal generators produce different plots to show the main characteristics of the signal facilitating understanding and analysis of the signals: PSD of the carrier signal, time domain plot of the carrier, PSD of the modulated signal, time domain plot of the modulated signal.

The output of the signal generator can be saved with an automatically generated name. The file contains the In-Phase and Quadrature components of the signal. In addition, the program provides the directory where the signal was saved as shown in Figure 60.

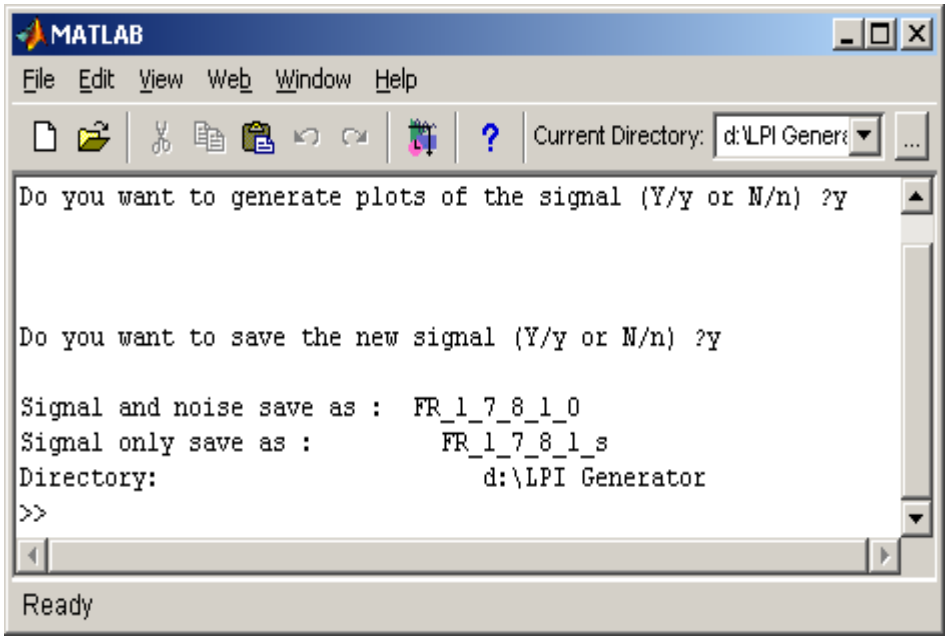


Figure 59 Polyphase-coded signal generator screen shot.

5. Costas code

The objective of this program is to generate LPI signals with Costas sequence coding. This program named **costas.m** and can be run from **LPIG.m** (main menu) or run directly. The execution of the program guides the user to the Costas code sub-menu, where many parameters can be adjusted: (see Figure 60 and Figure 61)

14. Amplitude of the carrier signal.
15. Sampling frequency (Hertz)
16. Signal to noise ratio (dB)
17. Cycles per frequency generated
18. Costas sequence: the user can pick one of two different Costas sequences.

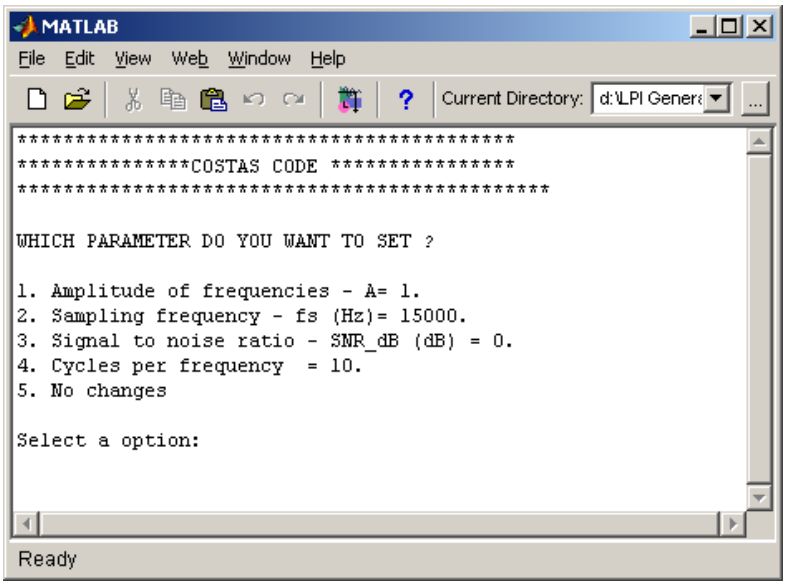


Figure 60 Costas code sub-menu.

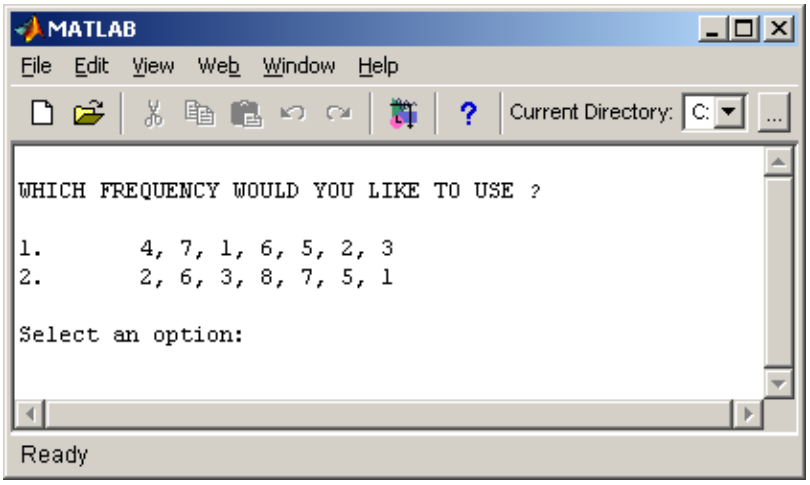


Figure 61 Costas sequence menu.

The Costas-coded signal generator produces the PSD of the resulting signal with noise and the PSD of the signal without noise facilitating understanding and analysis of signals. The output of the signal generator can be saved with an automatically generated name. The file contains the *I* and *Q* components of the signal. In addition, the program provides the directory where the signal was saved as shown in Figure 63.

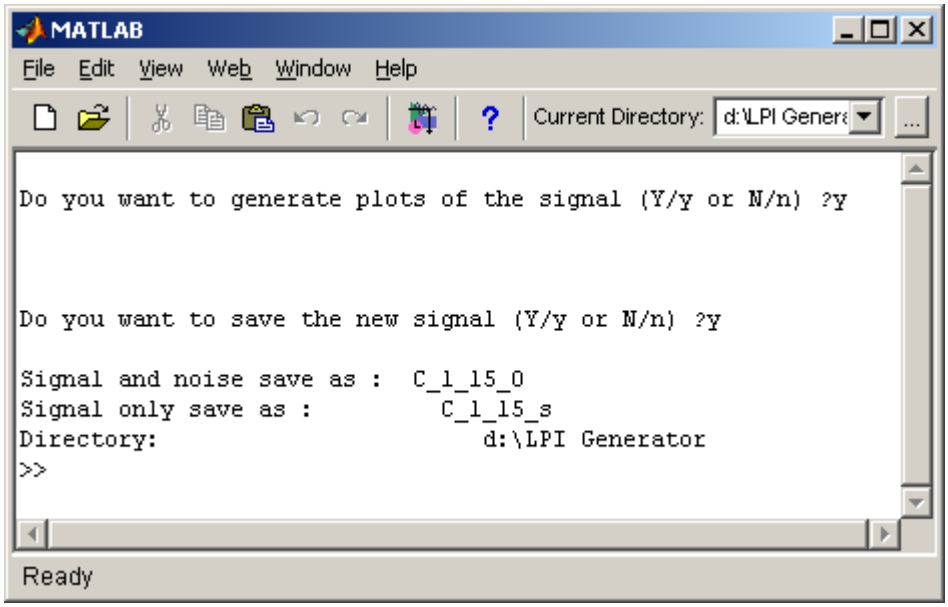


Figure 62 Costas signals generator.

In Chapter II, this document introduced the most important modulations employed to obtain LPI signals as well as the MATLAB® toolbox designed to generated such signals. Chapter III illustrates the proposed signal processing based on the use of a parallel filter bank and HOS. This chapter provides a theoretical background and the mathematical implementation of both the filter bank and the third-order cumulant estimators. Some examples are presented to the end of the chapter.

III. SIGNAL PROCESSING: PARALLEL FILTER ARRAYS AND HIGHER ORDER STATISTICS

A. OVERVIEW OF SIGNAL PROCESSING

In the early days of electronic warfare, the only option for an operator was to tune the radio across the entire available bandwidth to intercept threat signals. After detecting a signal, the operator could place noise jamming in the same detected carrier frequency with as much power as possible. The operator's ears and brain acted as the signal processing.

Today, an operator alone cannot efficiently intercept and analyze threat radar signals because of the multiple waveforms and modulations used. Modern receivers must detect, intercept, analyze and classify signals in very complex environments with noise, interference and multiple signals present. Therefore, LPI radar signals are especially designed to complicate the detection process to obscure the radar operations.

After the signals are detected, they must be classified into groups with similar characteristics. Some features are essential to discriminate one signal from another, such as carrier frequency, modulation type, rate and time, or angle and phase of arrival. Extracting all of these parameters leads to correctly identifying a signal. They can also be used to program an electronic attack. Many techniques have evolved lately for detecting and analyzing LPI signals. Most of these techniques are centered on time-frequency analysis which has many advantages over other periodogram techniques.

This thesis investigates a signal processing architecture based on the use of parallel filter arrays and HOS. A block diagram of this time-frequency approach is shown in Figure 63 . In this case study, the detector consists of an array of filters followed by higher-order cumulant estimators, envelope detectors, and a feature extraction process (done visually).

The LPIG signal generator constructs an input file in the form of I and Q. The input signal x is obtained by finding $x = I\text{-}jQ$. The input signal is then zero-padded in order to prevent circular convolution when later multiplying by the FFT of the band pass filter. The filter bank decomposes the introduced signal into sub-band signals with narrow frequency bands. Therefore, this filter bank was designed as a sine and cosine bank considering the real and imaginary part of the complex-valued impulse response of each bandpass filter.

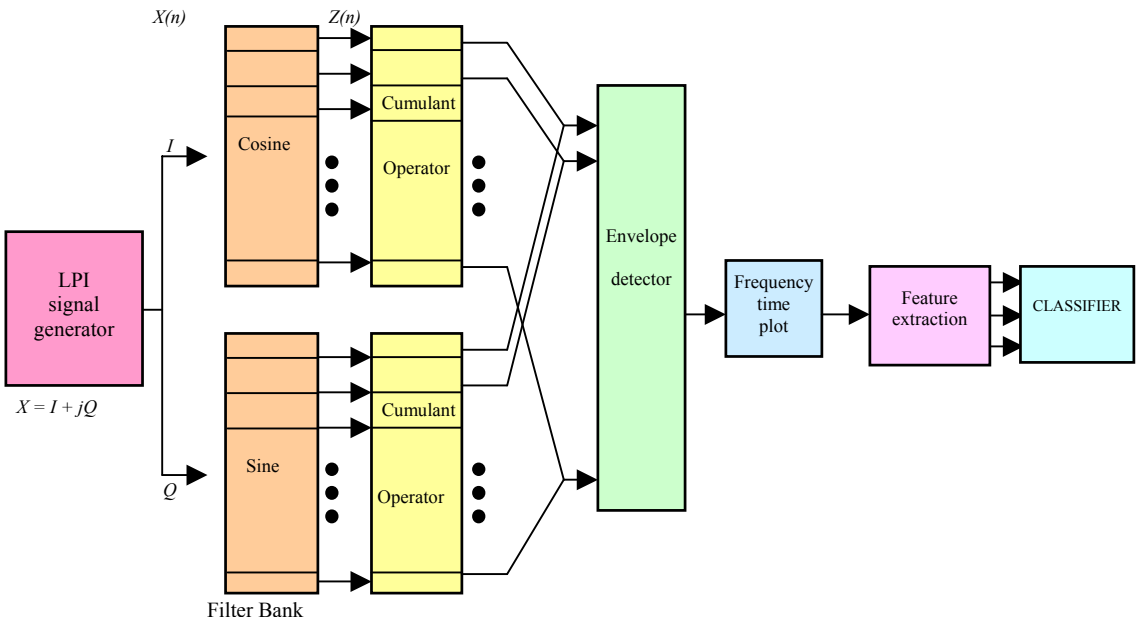


Figure 63 Overview of the proposed signal processing based on filter banks and HOS.

A third-order cumulant estimator (triple correlation) follows each sub-filter with the objectives of suppressing the additive Gaussian noise, extracting information due to deviations from Gaussianity and, at the same time, preserving the phase of the unknown signal. The outputs from each filter are combined using a envelope-approximation detector.

The analysis of the envelope-approximation detector output, using a visual feature extraction, provides the necessary characteristics to classify the signals. Some of the most important features given by the signal processing are the type of modulation (FMCW, Polyphase-coded signals, Costas code and PSK/FSK code), carrier frequency, modulation bandwidth, modulation time, phases codes, etc.

B. UNIFORM ARRAY OF FILTERS

1. Background of Filter Bank

The purpose of the filter bank is to split the frequency band of a signal into several sub-bands and to process the sub-band signals separately. The output of the sub-filters will be linearly independent, if the frequency functions of the sub-bands are almost non-overlapping [10] .

The impulse responses of a uniform filter bank with L subfilters are denotated $h_m(n)$, and the corresponding transfer functions are denotated $H_m(f)$. The impulse response is given by

$$h_m(n) = \frac{1}{L} h_o(n) e^{\frac{j2\pi(m-1)n}{L}} \tag{3.1.1}$$

where m = 1, 2, …, L and $h_o(n)$ is the impulse response of a prototype lowpass filter with frequency response of $H_o(f)$. The subfilters are constructed by the following relationship:

$$\left| \sum_{m=1}^{L} H_m(f) \right| = 1 \tag{3.1.2}$$

One prototype filter that fulfills this demand is

$$\begin{aligned} \frac{1}{2} H_o(f) &= \begin{cases} 1 - L|f|, & |f| < 1/L \\ 0, & \text{else} \end{cases} \\ \frac{1}{2} h_o(n) &= \frac{1}{L} \left(\frac{\sin \pi n / L}{\pi n / L} \right)^2 \end{aligned} \tag{3.1.3}$$

It will be approximated by a FIR filter of length N_f. From an implementation form of view, it is advantageous to relate the length N_f an the number L of subfilters according to $N_f = lL$ where l is an integer. An approximation of $|H_o(f)|$ in (3.1.3) with l =2 is shown in Figure 64 , the first three bandpass subfilters are depicted when the length of the approximation of $h_o(n)$ is N_f =128 and L=68. This approximation will only be slightly improved if l >2. Thus, for a given L. there will be only a minor improvement of the estimate by choosing l>2. The effect on the approximated impulse response is that its tails will be longer. It will, on the other

hand, increase the number of the numerical operations, which is proportional to N_f. The quality of the estimated signal will however be improved by increasing L. [10]

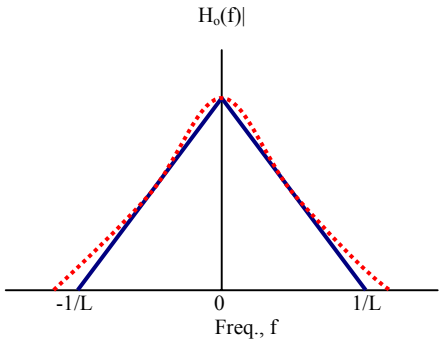


Figure 64 Magnitude function of the prototype low pass filter

2. The Design of a Uniform Filter Bank

When the impulse responses of a uniform filter bank with L sub-filters are $h_m(n)$, and the corresponding transfer functions are $H_m(f)$, the impulse responses are given by (3.1.1). Where $h_o(n)$ is the impulse response of a prototype low-pass filter with a frequency response of $H_o(f)$, $m = 1,...,L$ is the filter number , k is the gain, f_s is the sampling frequency and f is a frequency vector starting in 0 Hz with increments of $\frac{f_s}{length(input)}$, up to a length of $f_s - \frac{f_s}{length(input)}$. Another example of a lowpass filter is

$$h_o(n) = \frac{k \times \omega_p{}^2}{s^2 + s\frac{\omega_p}{q_p} + \omega_p{}^2} \tag{3.1.4}$$

where where $\omega_p = 2\pi \frac{f_s}{2L}$ and $s = 2\pi f j$. For this design the following parameters were used; $k = 1$ and $q_p = 0.707$ (quality factor) and f is the frequency vector built as

$$f = 0 : \frac{f_s}{length(input)} : f_s - \frac{f_s}{length(input)} \tag{3.1.5}$$

For example is the sampling frequency is 7 KHz, and the input number of samples is *N*=1225, then

$$f = 0 : \frac{7000}{1225} : 7000 - \frac{7000}{1225} \tag{3.1.6}$$

The impulse response is obtained using

$$h_o(n) = \Im^{-1}\{H_o(f)\} \tag{3.1.7}$$

The bandpass filter is creating by shifting the lowpass filter transfer function as

$$h_i(n) = h_o(n) \times e^{j2\pi f_i n} \tag{3.1.8}$$

where

$$f_i = f_i + \frac{f_s}{2L} \tag{3.1.9}$$

with $\frac{f_s}{2L}$ being the filter bandwidth.

To construct the filter bank, each bandpass filter is split into a Cosine and Sine filter

$$\begin{aligned} h_c(n) &= real\{h_i(n)\} \\ h_s(n) &= imag\{h_i(n)\} \end{aligned} \tag{3.1.10}$$

After constructing the cosine and sine filters in the time domain, the frequency domain is then

$$\begin{aligned} H_c(s) &= \Im\{h_c(n)\} \\ H_s(s) &= \Im\{h_s(n)\} \end{aligned} \tag{3.1.11}$$

The frequency domain filtering is then accomplished as

$$Y_c(s) = X(s)H_c(s)$$
$$Y_s(s) = X(s)H_s(s) \tag{3.1.12}$$

To get the time domain filter output

$$y_c(s) = \Im^{-1}\{Y_c(s)\}$$
$$y_s(s) = \Im^{-1}\{Y_s(s)\} \tag{3.1.13}$$

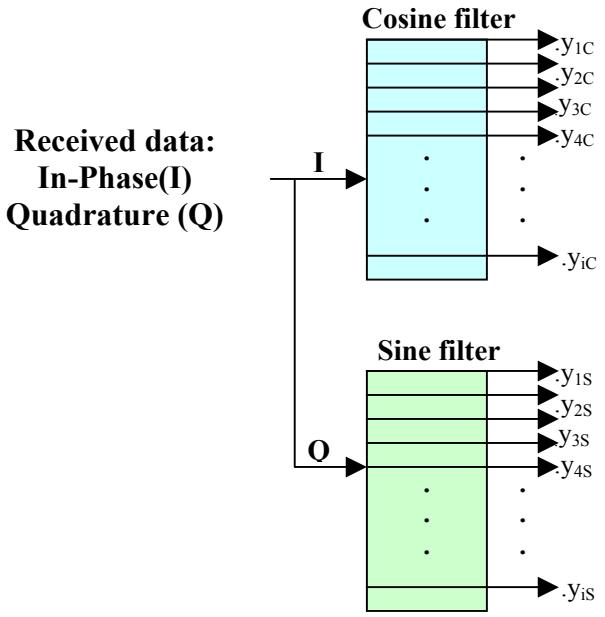


Figure 65 Final sine cosine filter bank.

3. Responses of the Parallel Filter Array to Different LPI Signals

With the objective of illustrating the response of the designed filter bank, the following section presents output for different LPI signals.

a. BPSK

Table 4 shows a BPSK signal with carrier frequency equal to 1 KHz, sampling frequency equal to 7 KHz, 7-bit Barker code and 5 cycles per bit. Figure 66 through Figure 68 show the output of the parallel filter arrays for 32 filters with signal only and SNR equal to 0 dB and –5 dB. The bandwidth of each filter is

$$B = \frac{f_s}{2L} = \frac{7000}{2(32)} = 109.375 \text{ Hz.}$$

Parameters	Values
Carrier Frequency	1000 KHz.
Sampling Frequency	7000 KHz.
Barker phase codes	7 bits
SNR	Signal only, 0 dB, -5 dB
Number of cycles per bit	5

Table 4 BPSK parameters.

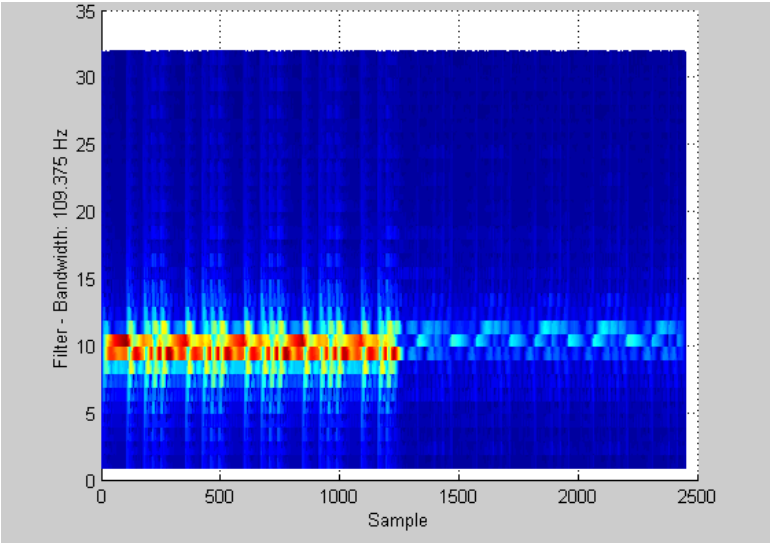


Figure 66 Response of the filter bank for a BPSK signal without noise, filter number versus samples with filter bandwidth = 109.375 Hz.

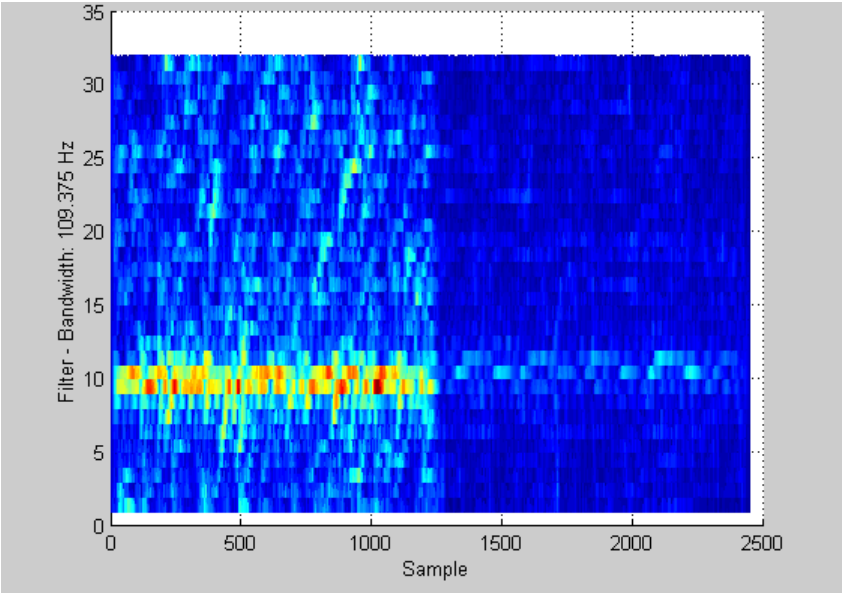


Figure 67 Response of the filter bank for a BPSK signal with SNR=0 dB.

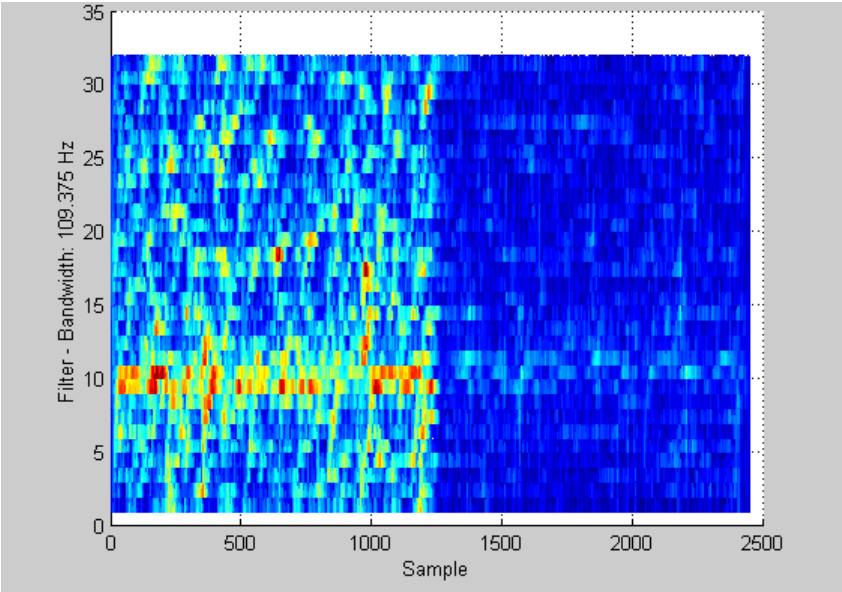


Figure 68 Response of the filter bank for a BPSK signal with SNR=-5 dB.

b. FMCW

Table 5 shows a FMCW signal with carrier frequency equal to 1 KHz, sampling frequency equal to 7 KHz, modulation bandwidth equal to 500 Hz and modulation period of 10 ms. Figure 69 through Figure 71 show the output of the parallel filter arrays for 32 filters with signal only and SNR equal to 0 dB and –5 dB. The bandwidth of each filter is $B = \frac{f_s}{2L} = \frac{7000}{2(32)} = 109.375$ Hz.

Parameters	Values
Carrier Frequency	1000 KHz.
Sampling Frequency	7000 KHz.
Modulation Bandwidth	500 Hz.
SNR	Signal only, 0 dB, -5 dB
Modulation period	10 ms

Table 5 FMCW parameters.

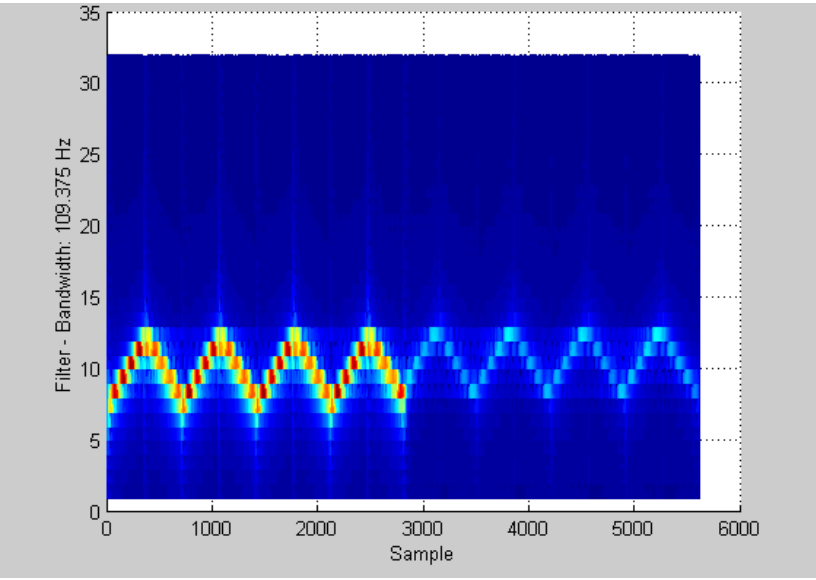


Figure 69 Response of the filter bank for a FMCW signal without noise.

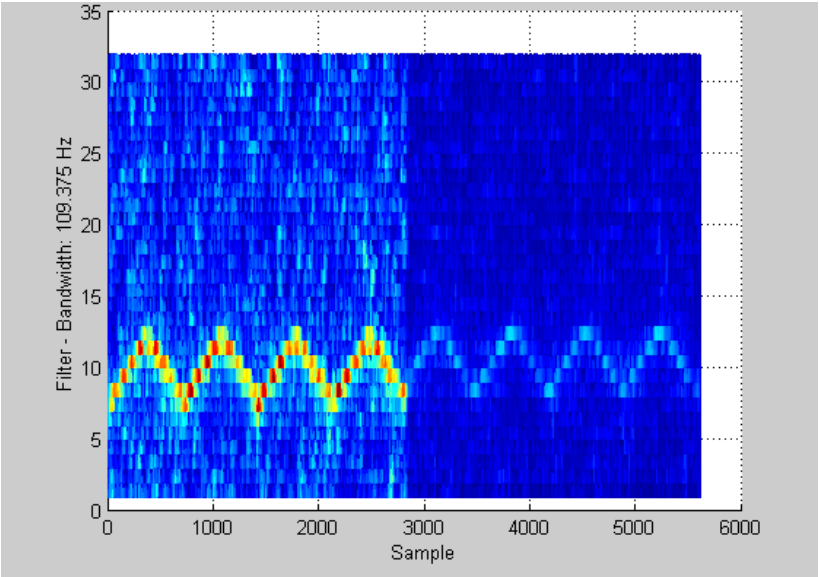


Figure 70 Response of the filter bank for a FMCW signal with SNR=0 dB.

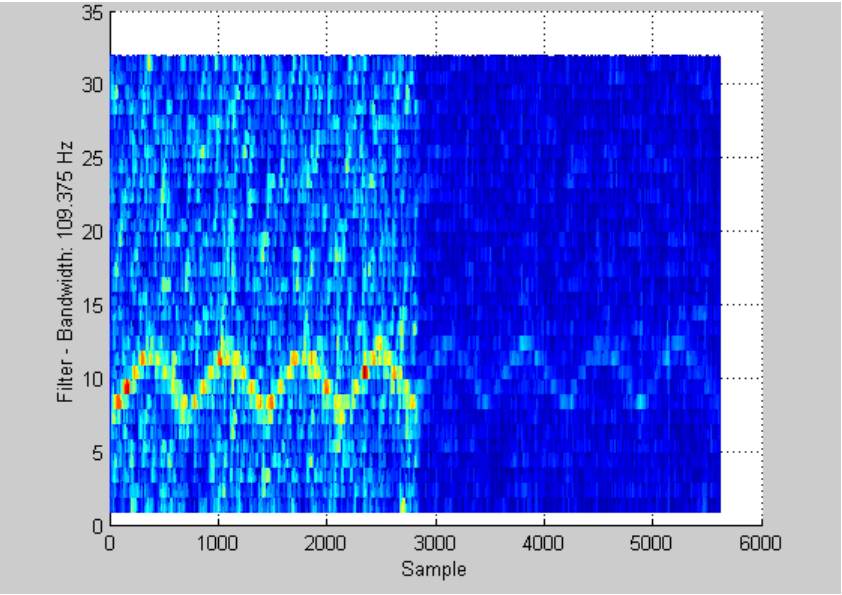


Figure 71 Response of the filter bank for a FMCW signal with SNR=-5 dB.

c. Polyphase code P4

Table 6 shows a P4 signal with carrier frequency equal to 1 KHz, sampling frequency equal to 7 KHz, 64 phases and 5 cycles per phase. Figure 72 through Figure 74 show the output of the parallel filter arrays for 32 filters with signal only and SNR equal to 0 dB and –5 dB. The bandwidth of each filter is $B = \frac{f_s}{2L} = \frac{7000}{2(32)} = 109.375$ Hz.

Parameters	Values
Carrier Frequency	1000 KHz.
Sampling Frequency	7000 KHz.
Number of phases	64
SNR	Signal only, 0 dB, -5 dB
Number of cycles per phase	5
Number of code period	5

Table 6 P4 parameters.

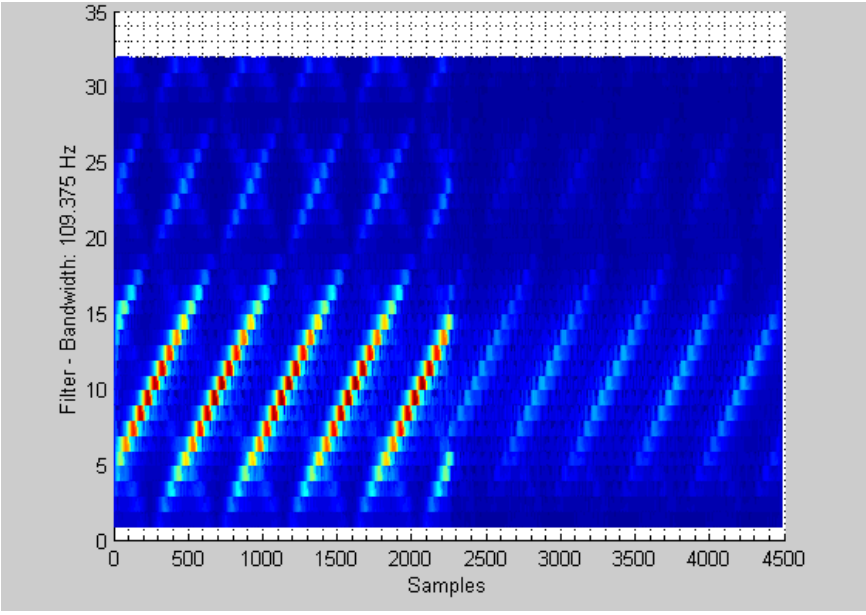


Figure 72 Response of the filter bank for a P4 signal without noise.

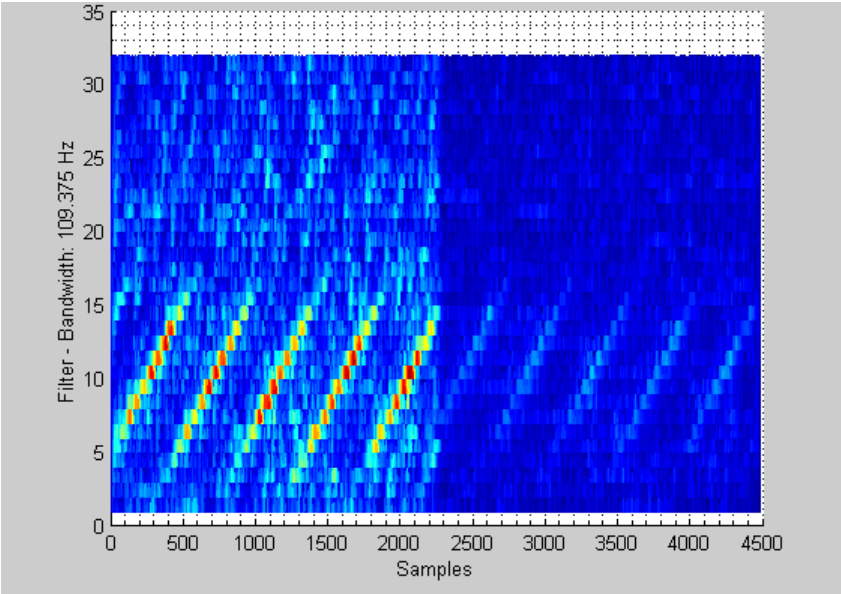


Figure 73 Response of the filter bank for a P4 signal with SNR=0 dB.

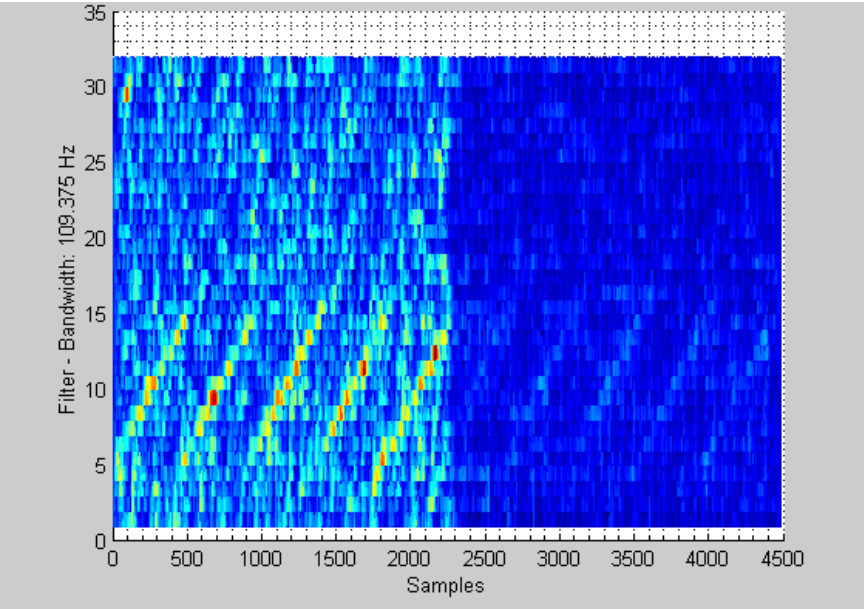


Figure 74 Response of the filter bank for a P4 signal with SNR=-5 dB.

d. Costas Code

Table 7 shows a P4 signal with carrier frequency equal to 1 KHz, sampling frequency equal to 7 KHz, 64 phases and 5 cycles per phase. Figure 75 through Figure 77 show the output of the parallel filter arrays for 32 filters with signal only and SNR equal to 0 dB and –5 dB. The bandwidth of each filter is $B = \frac{f_s}{2L} = \frac{15000}{2(32)} = 117.1875$ Hz.

Parameters	Values
Costas Sequence	4-7-1-6-5-2-3
Sampling Frequency	15000 Hz
Cycles per frequency	10
SNR	Signal only, 0 dB, -5 dB
Number of signals	5

Table 7 Costas code parameters.

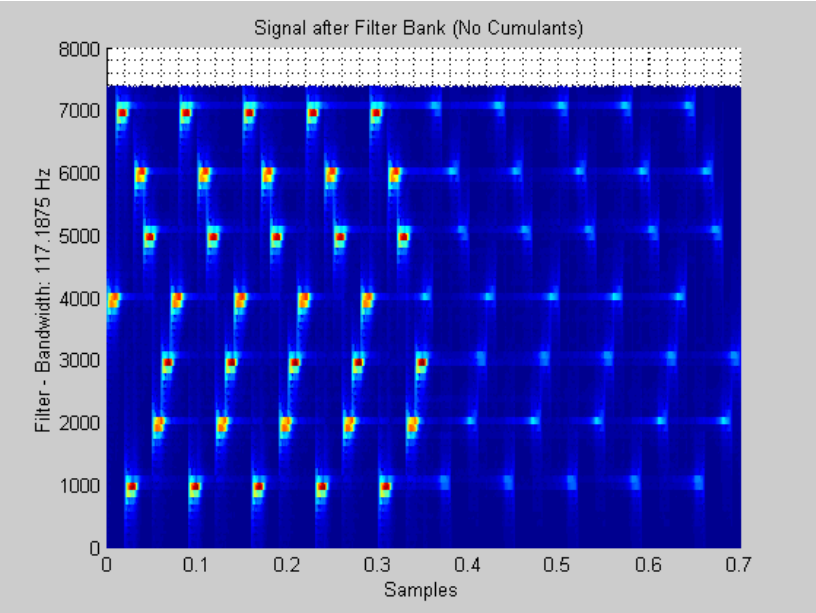


Figure 75 Response of the filter bank for a Costas signal without noise.

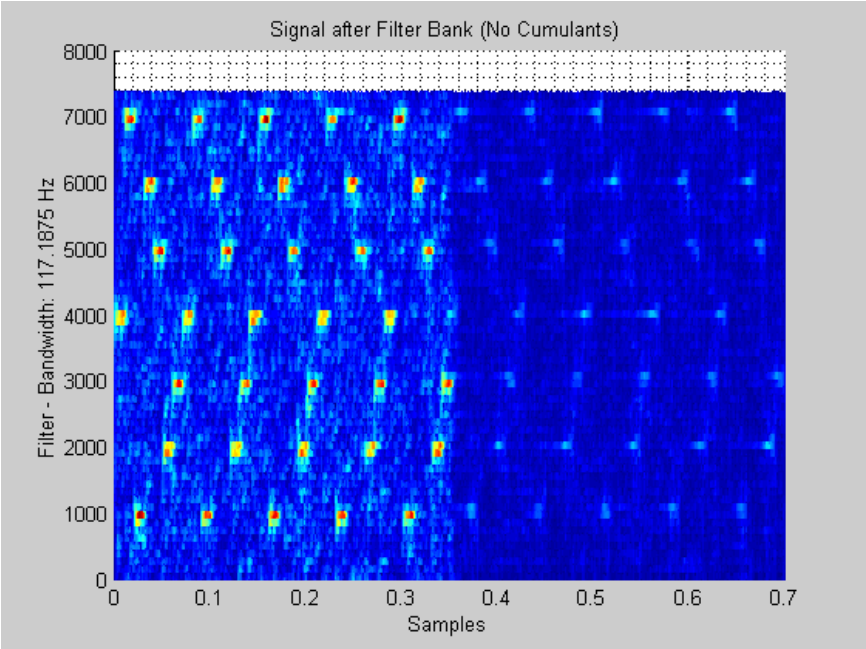


Figure 76 Response of the filter bank for a Costas signal with SNR= 0dB.

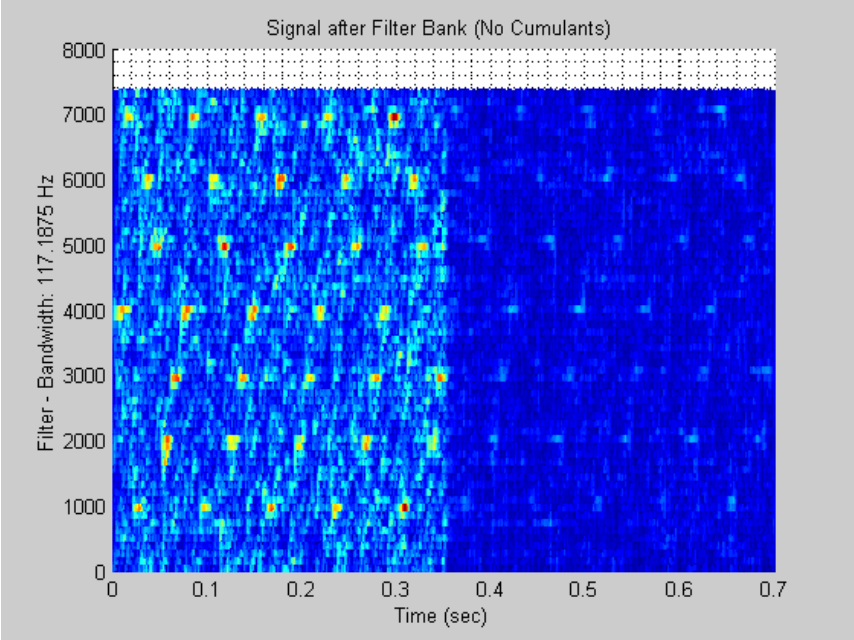


Figure 77 Response of the filter bank for a Costas signal with SNR= -5dB.

C. HIGHER ORDER STATISTICS (ESTIMATORS)

1. Introduction to Higher-Order Estimators

Recently, HOS have begun to find wide applicability in many fields, such as sonar, radar, plasma, physics, biomedicine, seismic data processing, image reconstruction, time-delay estimation, etc. These estimators are well known as cumulants. Their association with Fourier Transforms not only show the amplitude information but also can preserve phase information in a process.

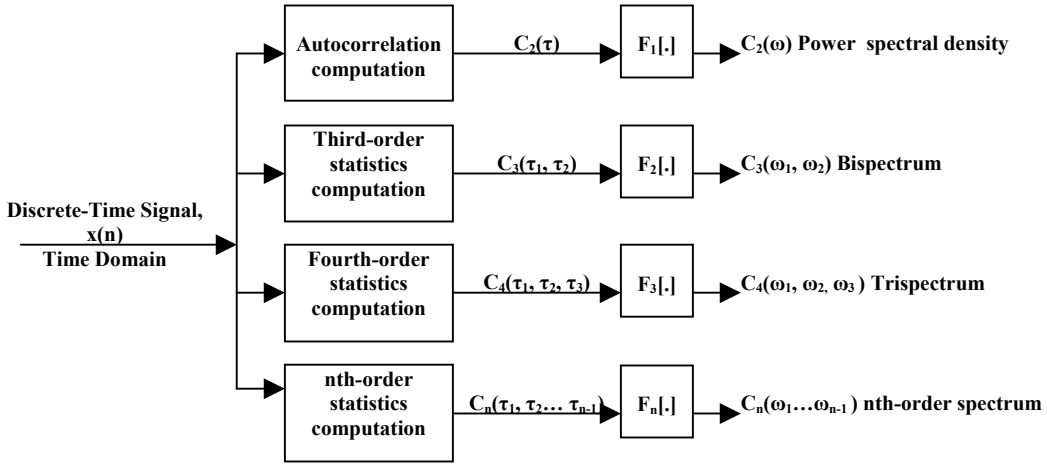


Figure 78 The higher-order spectral classification map of a discrete signal X(k). F[.] denotes n-dimensional Fourier Transform. (From[11])

In power spectrum estimation, the Fourier transform of the autocorrelation suppresses the phase relationship between frequency components. Power spectrums are phase blind. The information contained in the power spectrum is basically that which is present in the autocorrelation sequence; this is sufficient for the complete statistical description of a Gaussian signal. However, there are real situations where we must see beyond the power spectrum of a signal to extract information regarding deviation from Gaussianity and the presence of phase relations. LPI radar signals are examples of such a situation.

Cumulants, on the other hand, are blind to any kind of Gaussian processes. Cumulant-based methods improve SNR when signals are corrupted by Gaussian noise. Third-order cumulants are applicable when we are dealing with non-Gaussian or nonlinear systems; many

real world applications have this characteristic. The development of cumulants and polyspectra has paralleled the development of traditional correlation and its associated spectrum.

One of the most important motivations for the use of higher-order estimators is based on the property that for Gaussian signals only the whole cumulant spectra of order greater than two is identically zero. If a non-Gaussian signal is received along with additive Gaussian noise, transforming to a higher-order cumulant domain eliminates the noise. In general, cumulant spectra can become high SNR domains in which one may perform detection, parameter estimations or even an entire signal reconstruction.

If a random process is symmetrically distributed, then its third-order cumulants equal zero; therefore, for such a process it is necessary to use fourth-order cumulants. In addition, some processes have small third-order cumulants and much larger fourth-order cumulants. The biggest disadvantage of using HOS is that it requires longer data lengths than the correlation-based method. Longer data lengths are needed in order to reduce the variance associated with estimating the HOS from real data using sample-averaging techniques.

The idea behind the use of higher-order statistics in the proposed signal processing is precisely to evaluate the advantages of this method when detecting LPI (LPI) radar signals; to eliminate the noise added to the signal and to increase the SNR in later extracting the parameters needed for the correct classification.[12]

2. Mathematical implementation of HOS

In Sattar et al [9] they describe mathematically the application of HOS to estimate signals. The following development is an extract of their approach:

The output of each sub-filter is followed by a third-order cumulant estimator. A good approximation of a third-order cumulant estimator of a zero-mean signal is given by

$$\hat{C}_{3,z}(l_1,l_2;k)=\left[\frac{1}{(S_2-S_1+1)}\right]\sum_{n=S_1}^{S_2} z_k(n)z_k(n+l_1)z_k(n+l_2) \tag{3.2.1}$$

where l_1 and l_2 are the delays and

$$S_1=\max\{k-K,k-K-l_1,k-K-l_2\}$$
$$S_2=\min\{k+K,k+K-l_1,k+K-l_2\}$$
$$z_k(n)=\begin{cases} z(n)w(n-k) & k-K\le n\le k+K \\ 0 & \text{otherwise}\end{cases}$$

The length of the window $w(n)$ in the above equation is $2K+1$. For this implementation $K=2$ is used.

The estimates $\hat{C}_3(l_1,l_2;k)$ are reduced to one-dimensional (1-D) functions. These $\hat{C}_3(l,l;k)$ are suitable functions for detection, because the diagonal cumulant slices extract useful information, while the computational complexity remains modest. However, the cumulants will decrease rapidly with increasing l. Thus, for each frequency band, the nonlinear function

$$\rho_{3,i}(k)=\hat{C}_3(0,0;k)-\hat{C}_3(-1,1;k) \tag{3.2.2}$$

is used for the detection of LPI signal parameters.

As shown in [9] , this operator can be performed as

$$\rho_{3,i}(k)=\left(\frac{1}{2K+1}\right)\sum_{n}(s_{ik}(n)+v_{ik}(n))^3 -$$
$$\left(\frac{1}{2K+1}\right)\sum_{n}(s_{ik}(n)+v_{ik}(n))\times(s_{ik}(n-1)+v_{ik}(n-1))(v_{ik}(n+1)$$

$$\begin{aligned}\rho_{3,i}(k)=\left(\frac{1}{2K+1}\right)(&\sum_{n}(s_{ik}(n)+v_{ik}(n))^3-\sum_{n}(s_{ik}(n)s_{ik}(n-1)s_{ik}(n+1)\\&+s_{ik}(n)s_{ik}(n-1)v_{ik}(n+1)\\&+s_{ik}(n)v_{ik}(n-1)s_{ik}(n+1)\\&+s_{ik}(n)v_{ik}(n-1)v_{ik}(n+1)\\&+\mathrm{v}_{ik}(n)s_{ik}(n-1)s_{ik}(n+1)\\&+\mathrm{v}_{ik}(n)s_{ik}(n-1)v_{ik}(n+1)\\&+\mathrm{v}_{ik}(n)v_{ik}(n-1)s_{ik}(n+1)\\&+\mathrm{v}_{ik}(n)v_{ik}(n-1)v_{ik}(n+1)))\end{aligned}\qquad(3.2.3)$$

In order to separate the cross-terms, we can write the previous equation as

$$\begin{aligned}\rho_{3,i}(k)=\left(\frac{1}{2K+1}\right)(&\sum_{n}(s_{ik}^3(n)+v_{ik}^3(n)+3s_{ik}^2(n)v_{ik}^2(n)+3s_{ik}(n)v_{ik}^2(n))\\&-\sum_{n}(s_{ik}(n)s_{ik}(n-1)s_{ik}(n+1)\\&+s_{ik}(n)s_{ik}(n-1)v_{ik}(n+1)\\&+s_{ik}(n)v_{ik}(n-1)s_{ik}(n+1)\\&+s_{ik}(n)v_{ik}(n-1)v_{ik}(n+1)\\&+\mathrm{v}_{ik}(n)s_{ik}(n-1)s_{ik}(n+1)\\&+\mathrm{v}_{ik}(n)s_{ik}(n-1)v_{ik}(n+1)\\&+\mathrm{v}_{ik}(n)v_{ik}(n-1)s_{ik}(n+1)\\&+\mathrm{v}_{ik}(n)v_{ik}(n-1)v_{ik}(n+1)))\end{aligned}$$

$$\begin{aligned}\rho_{3,i}(k)=\left(\frac{1}{2K+1}\right)\times(&\sum_{n}(s_{ik}^3(n)+v_{ik}^3(n)-s_{ik}(n)s_{ik}(n-1)s_{ik}(n+1)\\&-v_{ik}(n)v_{ik}(n-1)v_{ik}(n+1))\\&+\ \text{cross-terms}\end{aligned}\qquad(3.2.4)$$

Now, the expression for $\rho_{3,i}(k)$ can be derived as a third-order function without having third-order harmonics. Assuming the output of the ith filter for the cosine bank is [9]

$$z_k^c(n) = A\cos(\omega_1 n + \psi i) \qquad k - K \le n \le k + K \tag{3.2.5}$$

then

$$\hat{C}_3^c(0,0;k) = \sum_{n=k-K}^{k+K} z_k^{c^3}(n) = \left(\frac{A^3}{4}\right) \sum_{n=k-K}^{k+K} \left[\cos 3(\omega_i n + \psi_i) + 3\cos(\omega_i n + \psi_i)\right] \tag{3.2.6}$$

and

$$\begin{aligned} \hat{C}_3^c(-1,1;k) &= \sum_{n=k-K}^{k+K} z_k^c(n) z_k^c(n-1) z_k^c(n+1) \\ &= \left(\frac{A^3}{4}\right) \sum_{n=k-K}^{k+K} (\cos(\omega_i n + \psi_i) + \cos 3(\omega_i n + \psi_i) \\ &\quad + 2\cos 2\omega_1 \cos(\omega_i n + \psi_i)) \end{aligned} \tag{3.2.7}$$

In the same way, if the output from the sine filter bank will be

$$z_k^S(n) = A\sin(\omega_1 n + \psi i) \qquad k - K \le n \le k + K \ , \tag{3.2.8}$$

then

$$\hat{C}_3^s(0,0;k) = \sum_{n=k-K}^{k+K} z_k^{s^3}(n) = \left(\frac{A^3}{4}\right) \sum_{n=k-K}^{k+K} \left[3\sin(\omega_i n + \psi_i) - \sin 3(\omega_i n + \psi_i)\right] \tag{3.2.9}$$

and

$$\begin{aligned} \hat{C}_3^s(-1,1;k) &= \sum_{n=k-K}^{k+K} z_k^s(n) z_k^s(n-1) z_k^s(n+1) \\ &= \left(\frac{A^3}{4}\right) \sum_{n=k-K}^{k+K} (\sin(\omega_i n + \psi_i) + \sin 3(\omega_i n + \psi_i) \\ &\quad + 2\cos 2\omega_1 \sin(\omega_i n + \psi_i)). \end{aligned} \tag{3.2.10}$$

Using 3.2.6, 3.2.7, 3.2.9 and 3.2.10, the complex expression of $\rho_{3,i}(k)$ would be

$$\begin{aligned}
\rho_{3,i}(k) &= C_3^c(0,0;k) + jC_3^s(0,0;k) \\
&\quad -[C_3^c(-1,1;k) + jC_3^s(-1,1;k)] \\
&= \frac{A^3}{4}\sum_{n=k-K}^{k+K}[e^{-j3(w_i n+\psi i)} + 3e^{-j3(w_i n+\psi i)}] \\
&\quad -\frac{A^3}{4}\sum_{n=k-K}^{k+K}[e^{j(w_i n+\psi i)} + e^{-j3(w_i n+\psi i)}] \\
&\quad + 2\cos 2\omega_i e^{j(w_i n+\psi i)} \\
&= \frac{A^3}{4}\sum_{n=k-K}^{k+K} 2(1-\cos 2\omega_i)e^{j(w_i n+\psi i)}] \\
&= \frac{A^3}{4}\sum_{n=k-K}^{k+K} \sin^2 \omega_i e^{j(w_i n+\psi i)}. \qquad (3.2.11)
\end{aligned}$$

3. Implementation of the Parallel Filter Arrays and HOS

To implement the filter bank and HOS described above, a set of programs was developed in MATLAB® version 6.1 running on a personal computer with the characteristics described in Table 8 .

Characteristic	
Processor (CPU)	Pentium IV
Speed	2 GHz.
RAM	1 Gigabyte
Operating system	Windows 2000 Professional
MATLAB®	Version 6.1

Table 8 Characteristics of software and hardware used in the simulation.

A set of two (02) programs were developed to perform all the tasks required:

a. Graphic User Interface (GUI).

The program called **hos_gui_1.m** was developed to call the main program. Using this GUI, the user only needs to input the right data in the fields provided and push the bottom to execute the program.

The main window requests information in four different fields:

- Data File: File name of the signal data correctly formatted in the I and Q components. The name does not need the extension .mat.
- Directory: Location of the file in any of the storage devices of the computer: Hard disk, floppy, zip drive, etc. If the exact location is not known, the letter of the device containing the file will be enough.
- Sampling Frequency: Frequency used to get samples from the input signal. It will be provided with the input signal.
- Number of Filter Bank: The input signal can be divided into 32, 64 or 128 sub-frequencies. This pop-up menu presents the number of filters that will be used to split the signal. The bandwidth of each filter is a function of the number of filters desired and the sampling frequency. If the number of filters increases and the sampling frequency stays constant, the bandwidth of each filter will decrease.

- Solve: This button executes the main program and, as a result, four plots are provided.

MATLAB® presents some flaws in executing and handling windows that must be known by the user. For example, every time the user enters new data in the fields, the "enter" key must be pushed, otherwise the new data is not accepted. In addition, the main program and the GUI generator must be in the same directory.

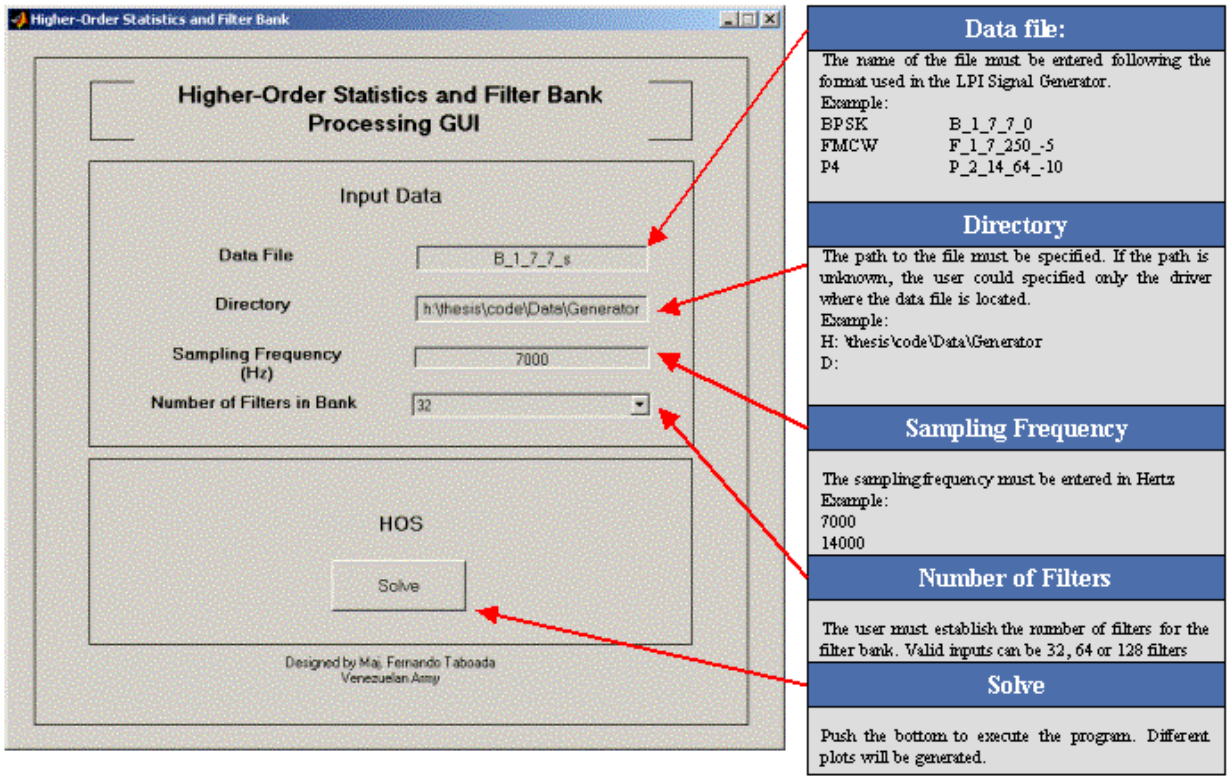


Figure 79 Graphic User Interface for the execution of the filter bank and calculation of the HOS.

b. Main Program.

The main program, **taboada_hos_gui.m**, performs all the calculations needed to implement the filter bank and execute the algorithms necessary to obtain the higher-order estimators. This file is in Appendix B.

This program generates four different plots presenting characteristics of the resulting signal after the filter bank and before the higher-order estimators.

- Frequency-Time plot of the signals after the Filter Bank
- Frequency-Time plot of the signal after the Higher-Order estimators
- Frequency-Amplitude plot
- Filter-Amplitude plot

c. Resulting Plots for Different LPI Radar Signals

(1) FMCW

Table 9 shows a FMCW signal with carrier frequency equal to 1 KHz, sampling frequency equal to 7 KHz, modulation bandwidth equal to 500 Hz and modulation period of 10 ms. Figure 80 shows the output of HOS for signal only and SNR equal to 0 dB.

Parameters	Values
Carrier Frequency	1000 KHz.
Sampling Frequency	7000 KHz.
Modulation Bandwidth	500 Hz.
SNR	Signal only and 0 dB
Modulation period	10 ms

Table 9 FMCW parameters.

This figure presents the output after the parallel filter arrays, the output after HOS and two different views, the amplitude-frequency view and the amplitude-filter view.

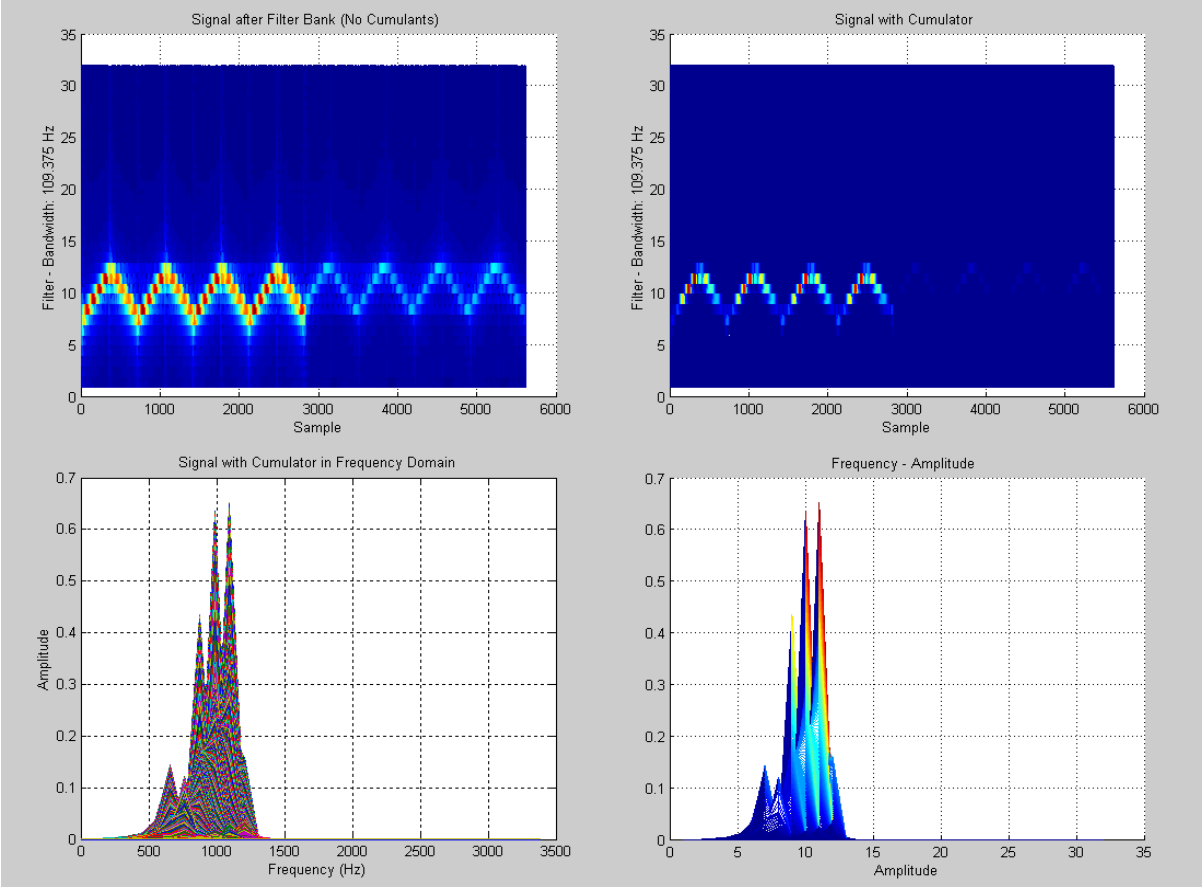


Figure 80 Resulting plots from the signal processing: before and after HOS and two differents views, amplitude-frequency and amplitude-filters (signal only).

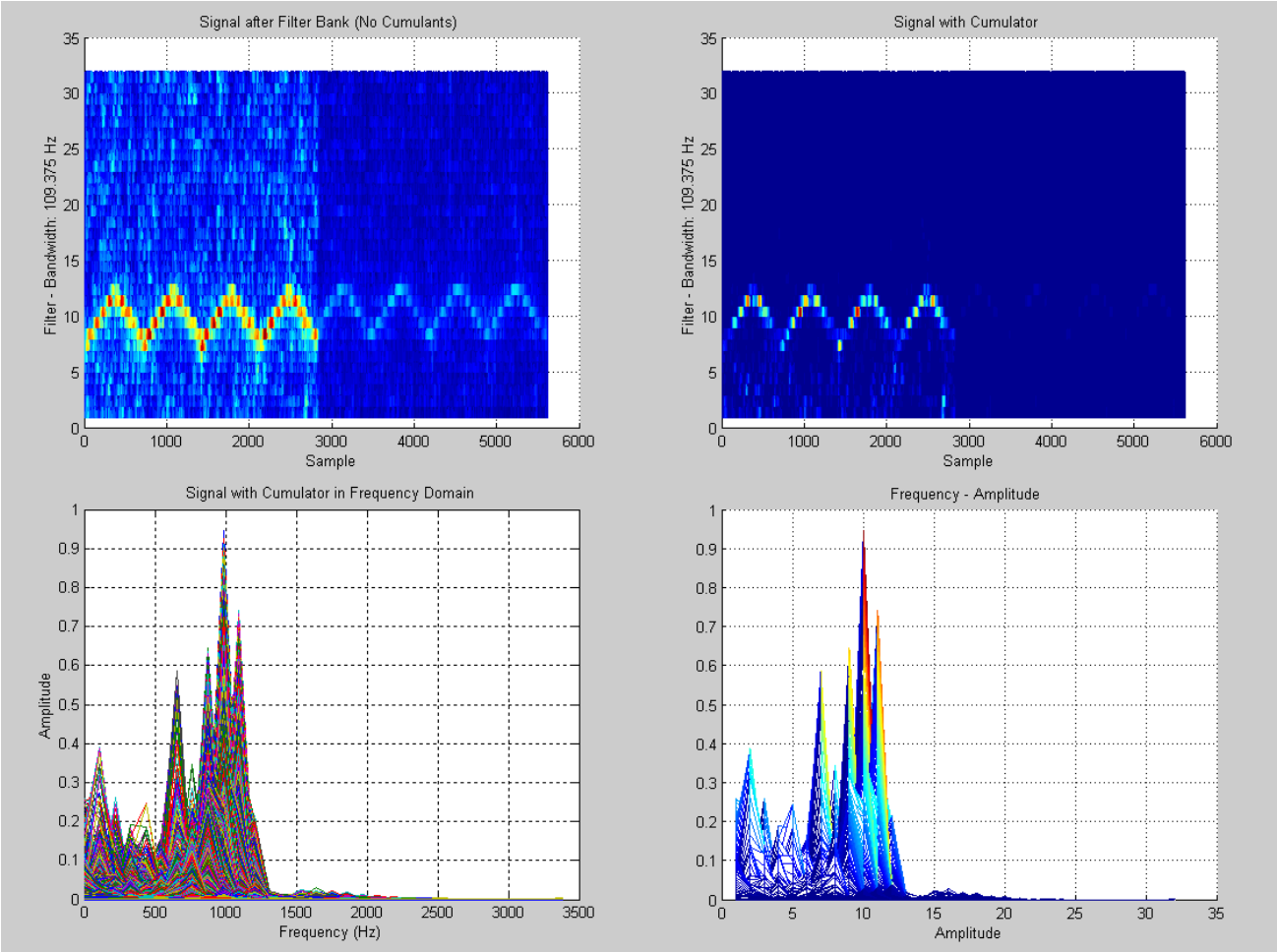


Figure 81 Resulting plots from a FMCW signal: before and after HOS and two differents views, amplitude-frequency and amplitude-filters (SNR = 0 dB).

(2) Polyphase P4

Table 10 shows a P4 signal with carrier frequency equal to 1 KHz, sampling frequency equal to 7 KHz, 64 phases and 5 cycles per phase. Figure 82 and Figure 83 shows the output of HOS for signal only and SNR equal to 0 dB.

Parameters	Values
Carrier Frequency	1000 KHz.
Sampling Frequency	7000 KHz.
Phases	64.
SNR	Signal only, -5 dB
Number of cycles per phase	5

Table 10 P4 parameters.

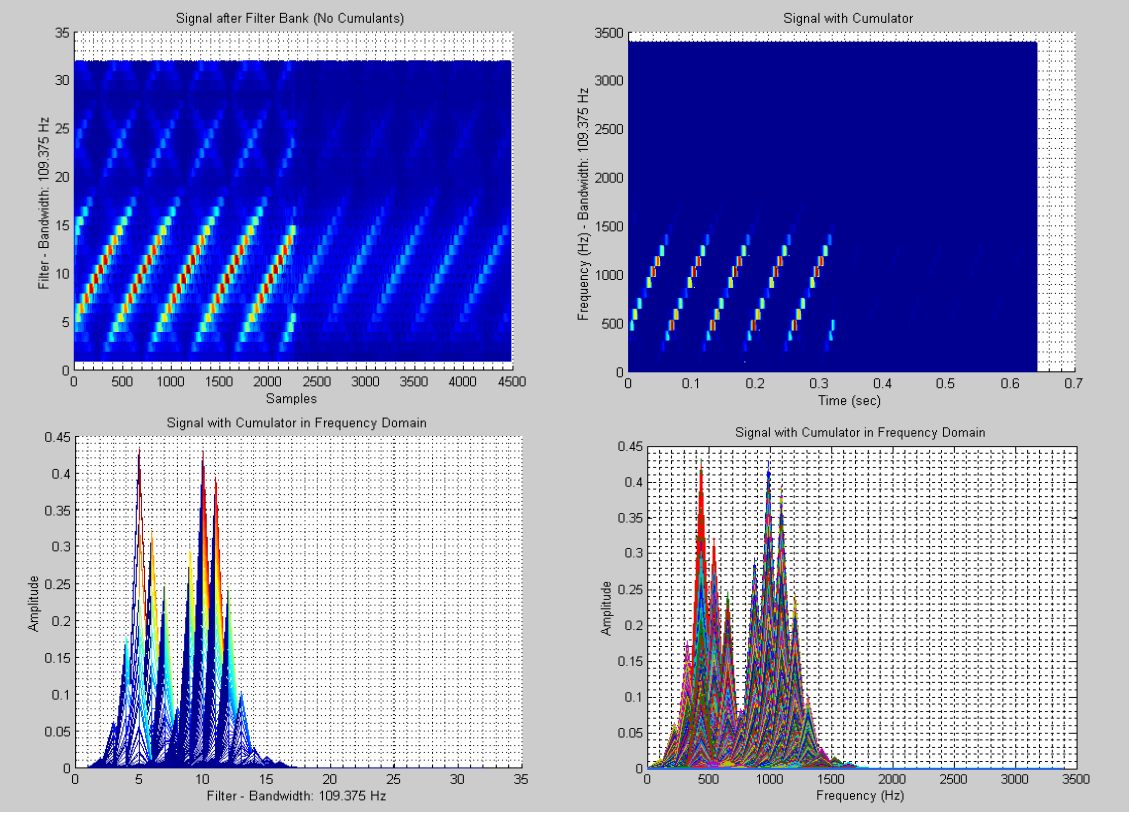


Figure 82 Resulting plots from a P4 signal : before and after HOS and two different views, amplitude-frequency and amplitude-filters (signal only).

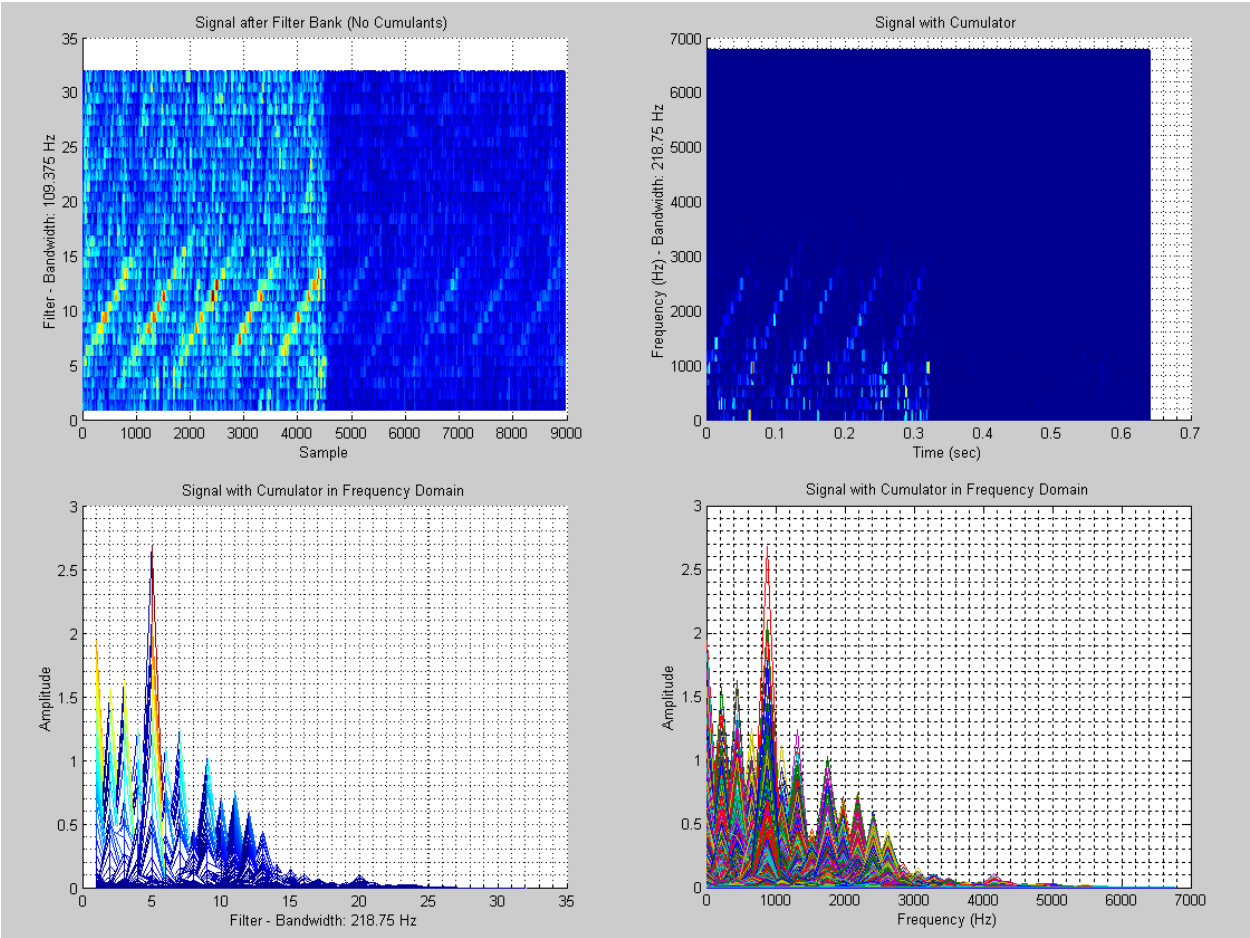


Figure 83 Resulting plots from a P4 signal : before and after the HOS and two different views, amplitude-frequency and amplitude-filters (SNR = -5 dB).

Chapter III introduced the proposed signal processing based on the use of a parallel filter bank and HOS. This chapter provided a theoretical background and the mathematical implementation of both the filter bank and the third-order cumulant estimators. Chapter IV shows the analysis of a test signal matrix designed to test the effectiveness of the proposed signal processing.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ANALYSIS OF RESULTS

Using the LPI signal generator described in Chapter II, different inputs signals were designed to demonstrate the performance of the signal processing. For each modulation, some of the parameters were varied to create a complete set of diverse signals where their most important characteristics are demonstrated. The analysis of the signals and their results are presented in the following order:

a. A table showing the parameters of the input signal and the extracted parameters after the processing.
b. A tutorial on how to calculate the parameters .
c. Graphical description of the input signal: PSD, Time domain plot, PAF and cuts along the 0 Doppler and the 0 delay axis.
d. Resulting plots after the HOS signal processing: Resulting signal after the parallel filter arrays, resulting signal after HOS, amplitude-filter plot after HOS and amplitude-frequency plot after HOS.
e. The most important parameters are indicated in plots with lines, arrows and comments.

The carrier frequency is kept constant for all the input signals in this research (f_c=1000 Hz) to minimize the number of signals to analyze. Additionally, the sampling frequency is selected to be seven times the carrier frequency (f_s = 7000 Hz). Only in the case of Costas-coded signals, the sampling frequency is set to 15000 Hz since the highest frequency in Costas set is 7000 Hz.

A. TEST SIGNAL

One-frequency signal and a two-frequency signal are generated and analyzed to demonstrate the response of the parallel array of filters and the application of third-order cumulant estimators. Table 11 describes two different signals.

No	Signal file	Carrier Frequency	Sampling Frequency	SNR
1	T 1 7 1 s	1000	7000	-
2	T 12 14 2 s	1000, 2000	14000	-

Table 11 Test signals.

This section first describes the most important characteristics of the input signal providing the PSD. Then, the resulting plots after the proposed signal processing are presented and analyzed in detail.

As a result of selecting 64 filters in the parallel filter arrays, the bandwidth of each sub-filter is 54.68 Hz. This value is obtained by

$$B = \frac{f_s}{2L} \tag{4.1.1}$$

where f_s is the sampling frequency in Hz and L is the number of filters in bank. The simplicity of the results obtained by processing these two signals can help understand better what we expect when processing more complex signal such as BPSK, FMCW or polyphase-coded signals.

1. Single tone

Table 12 describes a single tone signal with carrier frequency 1KHz and sampling frequency 7 KHz which is analyzed using the parallel filter arrays and HOS. Figure 84 presents the PSD of the input signal, showing the power distribution of the signal in each of the frequencies. As a result of a single tone, all the power is concentrated in the carrier frequency.

Single tone – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
SNR	Signal only	-	

Table 12 Test Signal with one carrier.

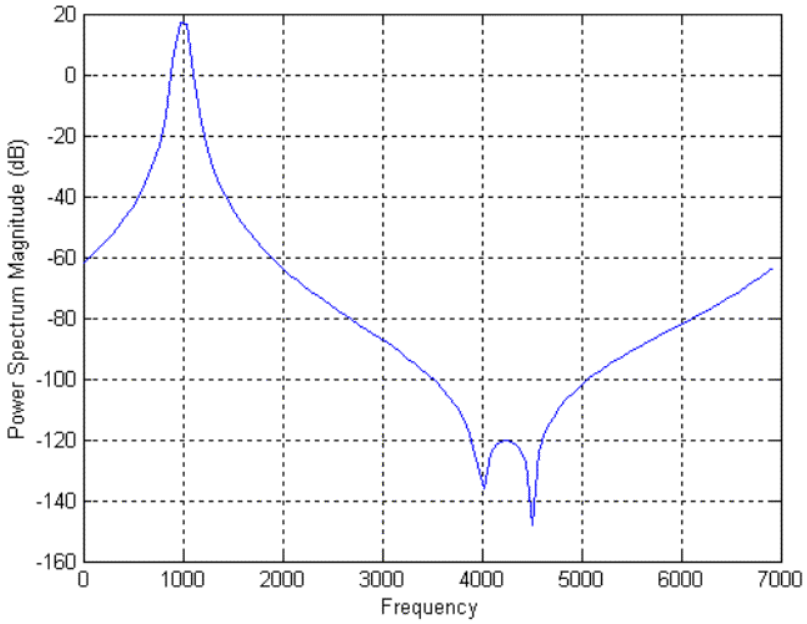


Figure 84 Single tone PSD.

The response of the parallel filter arrays is presented in Figure 85 . The objective of the filter bank is to decompose the input signal into sub-band frequency bands. This plot provides a good frequency-time representation of the signal.

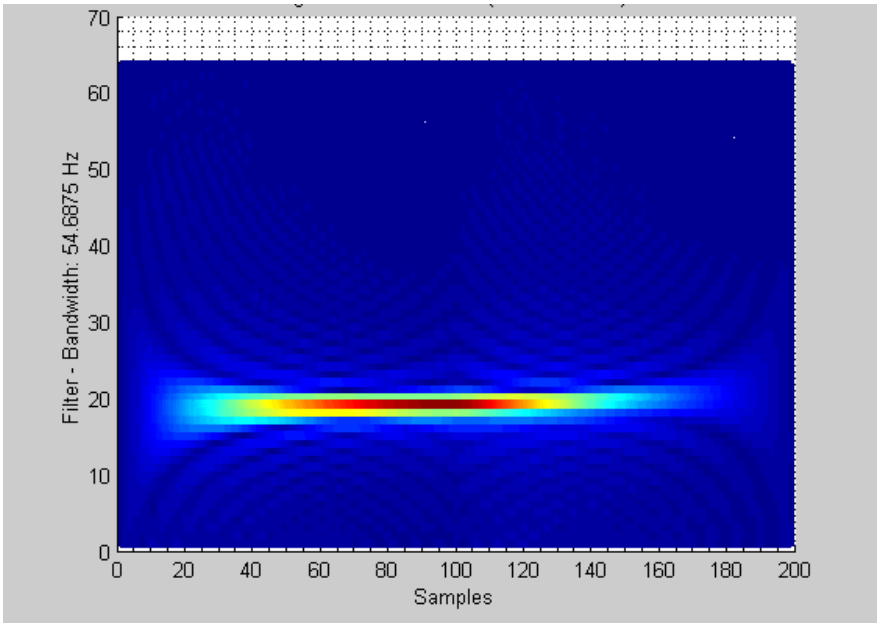


Figure 85 Single Tone output of the parallel filter arrays.

Then, each sub-band filter is followed by a third-order cumulant estimator to suppress the white Gaussian noise. Since any of the test signals don't have noise added, the result from after applying HOS is not very different to the previous step.

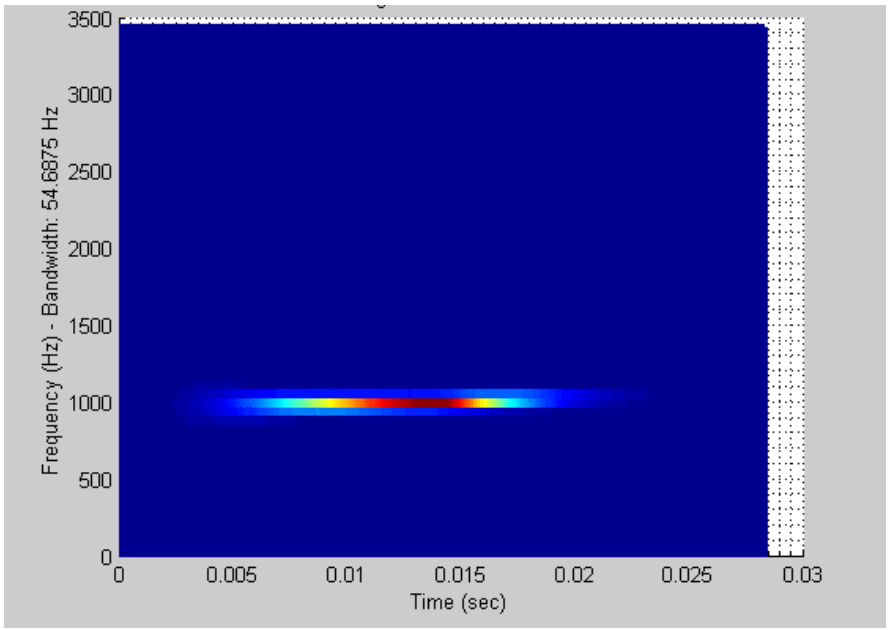


Figure 86 Single tone output after HOS.

Figure 87 shows an amplitude-frequency view of the resulting figure after HOS. This plot reveals the center frequency of the signal (carrier) and the bandwidth occupied by a single tone.

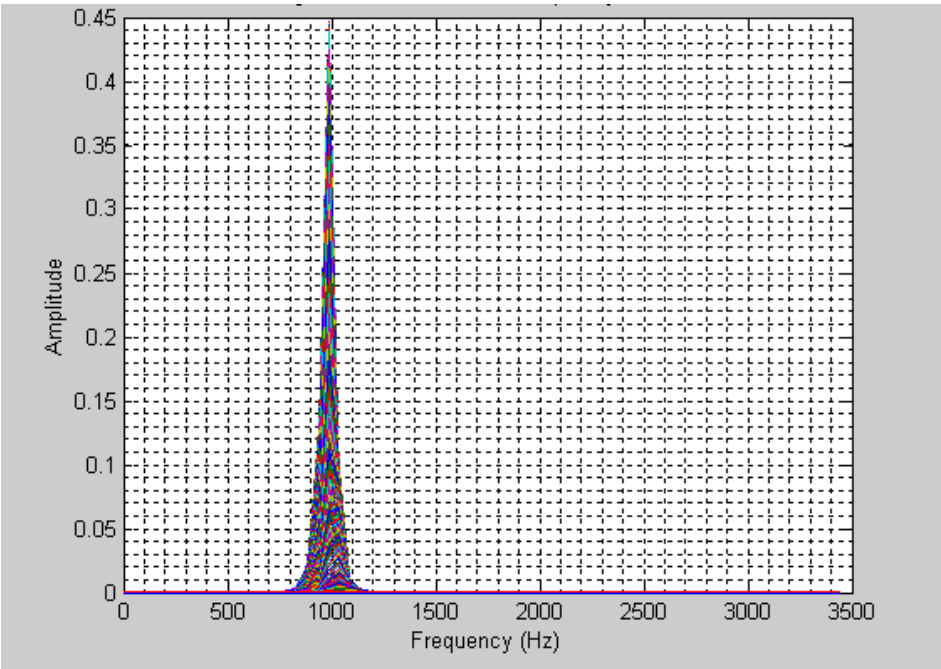


Figure 87 Single tone amplitude-frequency plot.

2. Two-frequency tone

Figure 88 presents the PSD of a two-frequency signal with carrier frequencies of 1 KHz and 2 KHz, and sampling frequency equal to 14 KHz. The power is distributed in the two carrier frequencies.

Two-frequency signal – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000, 2000	1000, 2000	
Sampling frequency (Hz)	14000	14000	Given
SNR	Signal only	-	

Table 13 Test Signal with two carrier frequencies.

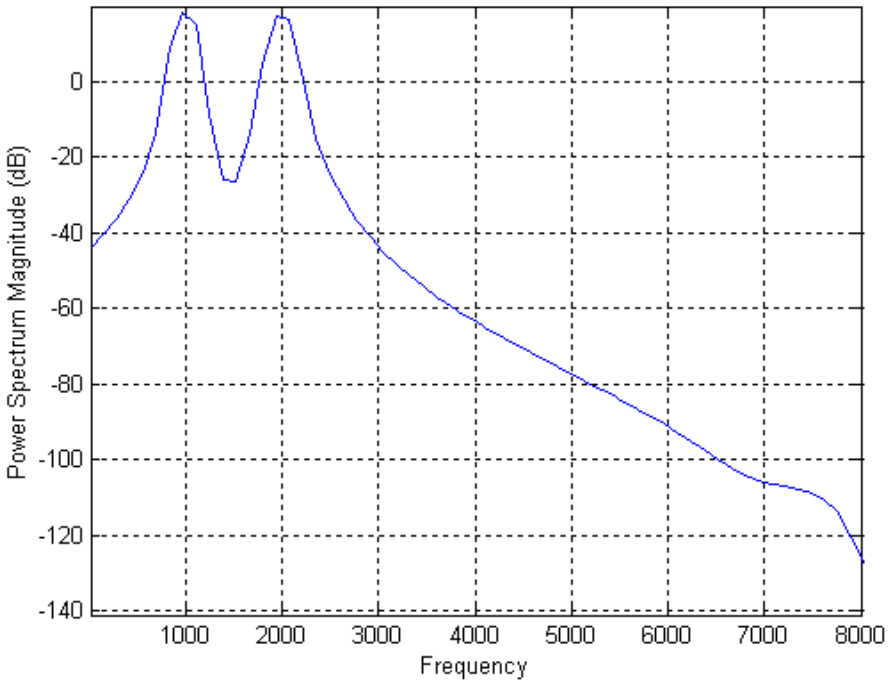


Figure 88 Two-frequency tone PSD.

Figure 89 shows the output of the parallel filter arrays. The input signal is decomposed in sub-band frequencies to provide a complete frequency-time description. The carrier frequencies are located in filters 9 and 19. It means that the detected carrier frequencies are 984.33 ± 109.37 Hz and 2078.03 ± 109.37 Hz. The error is related to the bandwidth of each sub-filter, which is calculated by equation (4.1.1) and it depends on the sampling frequency and the number of filters in bank.

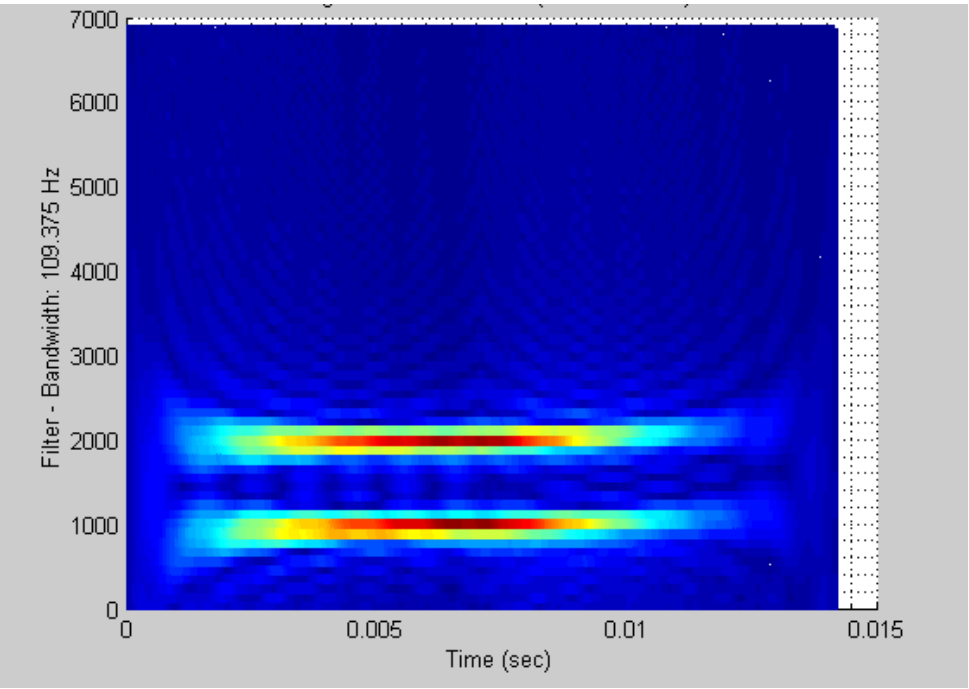


Figure 89 Two-frequency signal output of the parallel filter arrays.

Because this signal doesn't have noise added the third-order estimators applied to each sub-filter don't produce an important change in the previous resulting plot. The performance of the HOS will be observed later when analyzing more complex noisy signals.

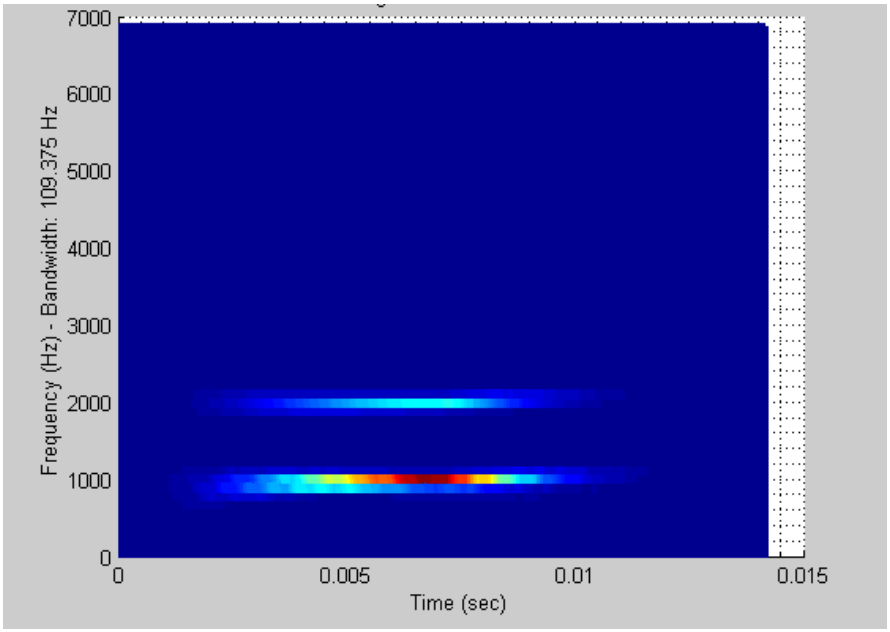


Figure 90 Two-frequency signal output after HOS.

The amplitude-frequency view provided in Figure 91 is and important representation of the signal in the frequency domain where all the frequency components can be observed. In the case of a two-frequency signal, two spikes appear centered at the carrier frequencies.

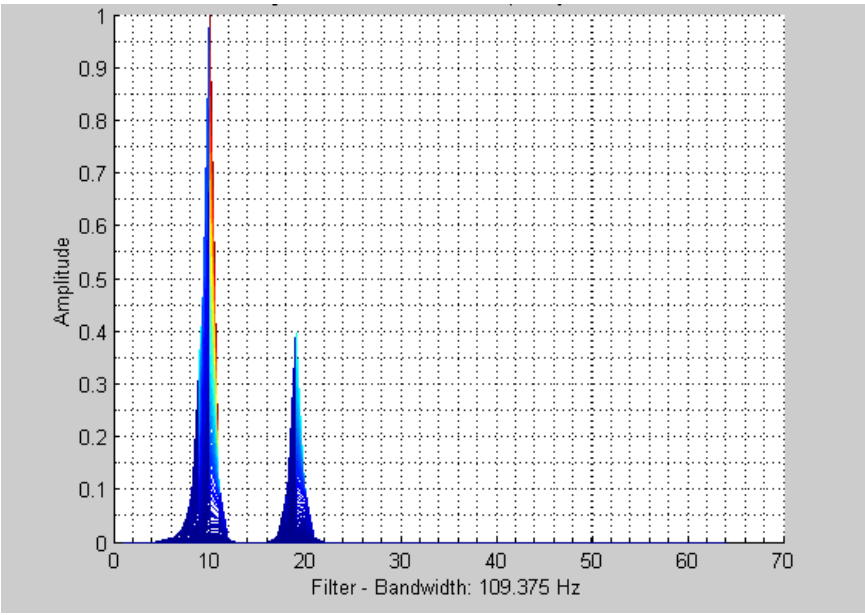


Figure 91 Two-frequency signal amplitude-frequency plot after HOS.

B. BPSK

Binary Phase Shift-Keying (BPSK) is modulation technique that has proven to be extremely effective in communication and radar systems. Even though BPSK is not a technique employed in LPI radar, this technique is an excellent test signal in evaluating the performance of the proposed signal processing.

In BPSK modulation, the phase of the frequency carrier is shifted 180 degrees in accordance with the digital bit stream. The digital coding scheme used is called Non-Return-to-Zero (NRZ-M). A "one" causes a phase-transition, and a "zero" does not produce a transition.

As shown in Table 14 a set of twelve signal is analyzed to evaluate the performance of the parallel filter arrays detecting and identifying the most important parameters in the input signal. Some parameters are varied in the input signal, such as the number of bits in Barker code, the number of cycles per bit and SNR. Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	Signal file	Carrier Frequency	Sampling Frequency	Bits in Barker code	Cycles per bit	SNR
1	B 1 7 7 1 s	1000	7000	7	1	-
2	B 1 7 7 1 0	1000	7000	7	1	0
3	B 1 7 7 1 -6	1000	7000	7	1	-6
4	B 1 7 11 1 s	1000	7000	11	1	-
5	B 1 7 11 1 0	1000	7000	11	1	0
6	B 1 7 11 1 -6	1000	7000	11	1	-6
7	B 1 7 7 5 s	1000	7000	7	5	-
8	B 1 7 7 5 0	1000	7000	7	5	0
9	B 1 7 7 5 -6	1000	7000	7	5	-6
10	B 1 7 11 5 s	1000	7000	11	5	-
11	B 1 7 11 5 0	1000	7000	11	5	0
12	B 1 7 11 5 -6	1000	7000	11	5	-6

Table 14 Matrix of input signals for BPSK.

1. BPSK, 7-bit Barker code, 5 cycles per phase and signal only

Table 15 describes a BPSK signal with carrier frequency 1 KHz, sampling frequency 7 KHz, 7-bit Barker code, and 5 cycles per bit. By increasing the cycles per phase to 5, the bandwidth is reduced to a fifth of its original bandwidth, as shown in equation(4.1.3) . In the same way, the modulation period is also affected because it depends on the cycle per phase.

BPSK – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
Bandwidth (Hz)	200	218.64	
Bits per Barker code	7	-	
SNR (dB)	Signal only	-	
Cycles per bit	5	5	
Code Period (ms)	35	35	

Table 15 BPSK, 7-bit Barker code, 5 cycles per phase and signal only.

The modulation period of the BPSK signal is

$$t_c = \frac{(\text{Cycles/bit})(\text{Barker bits})}{f_c} = \frac{(5)(7)}{1000} = 35 \text{ ms} \tag{4.1.2}$$

The bandwidth of the signal depends on the cycles per bit (or chirp) as

$$B = \frac{f_c}{\text{Cycles per bits}} = \frac{1000 \text{ Hz}}{5 \text{ cycles per bit}} = 200 \text{ Hz} \tag{4.1.3}$$

The PSD for the BPSK signal without noisy is presented in Figure 92 . This plot provide a description of the distribution of the power in the frequency domain confirming results in (4.1.2) and (4.1.3).

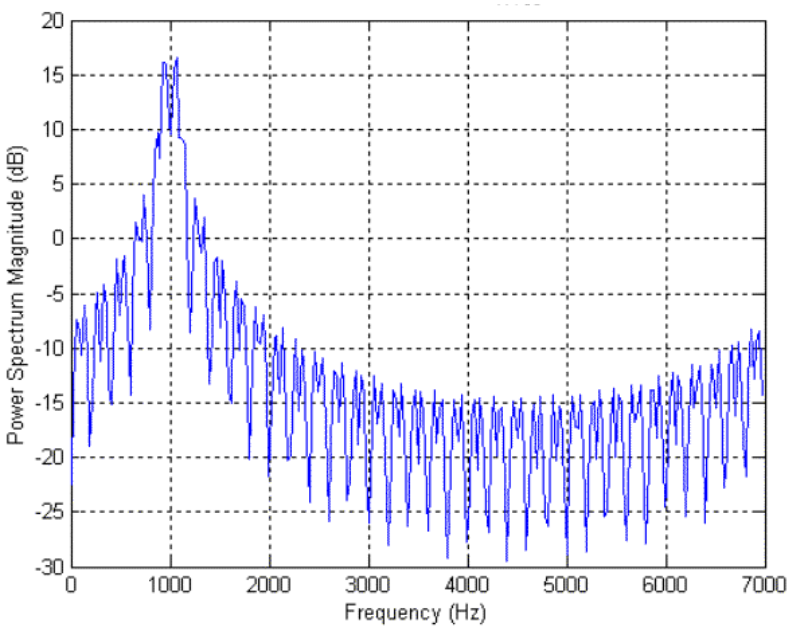
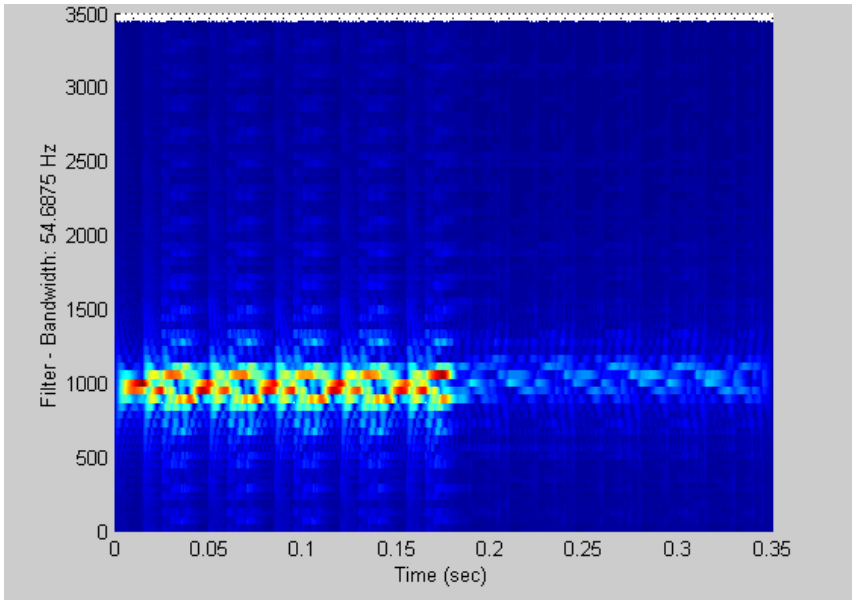


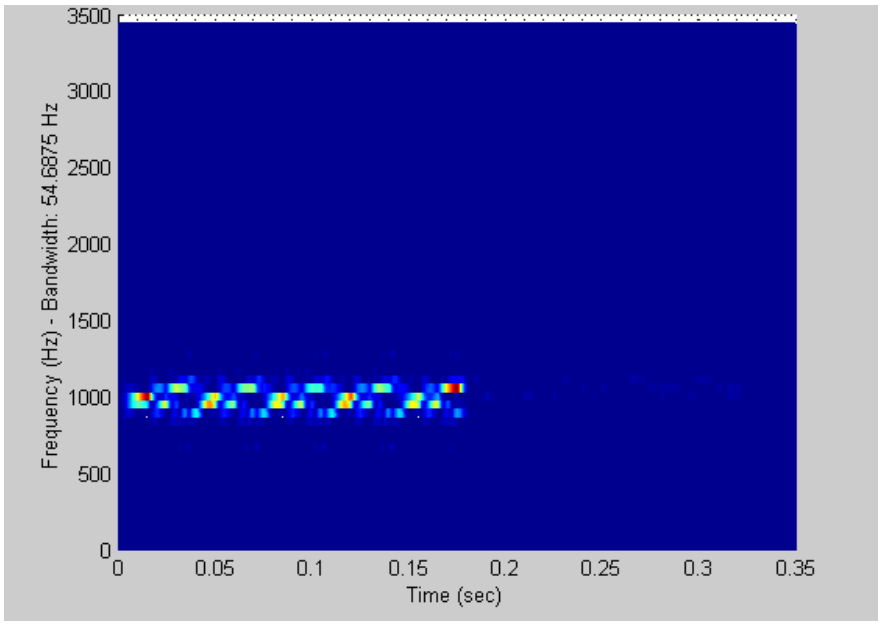
Figure 92 BPSK, 7-bit Barker code, 5 cycles per phase, signal only PSD.

Figure 93 shows the resulting plots obtained by analyzing the signal with the proposed signal processing. Figure 93 (a) illustrates the output of the parallel filter arrays. The input signal is divide into small frequencies to provide a complete time-frequency description. Figure 93 (b) corresponds to the resulting plot after the third-order cumulant estimators. Due to the lack of noise in this signal, the only effect of the HOS is a small degradation of the signal.

Figure 94 makes a zoom in the resulting plot after HOS to estimate carrier frequency, bandwidth and code period of the processed signal. Figure 95 shows an amplitude-frequency view of the resulting signal after HOS. This plot can be used to examine the distribution of the different frequency component the signal in the filter bank.



(a)



(b)

Figure 93 BPSK, 7-bit Barker code, 5 cycles per phase and signal only (a) Output of the parallel filter arrays (b) Output after HOS.

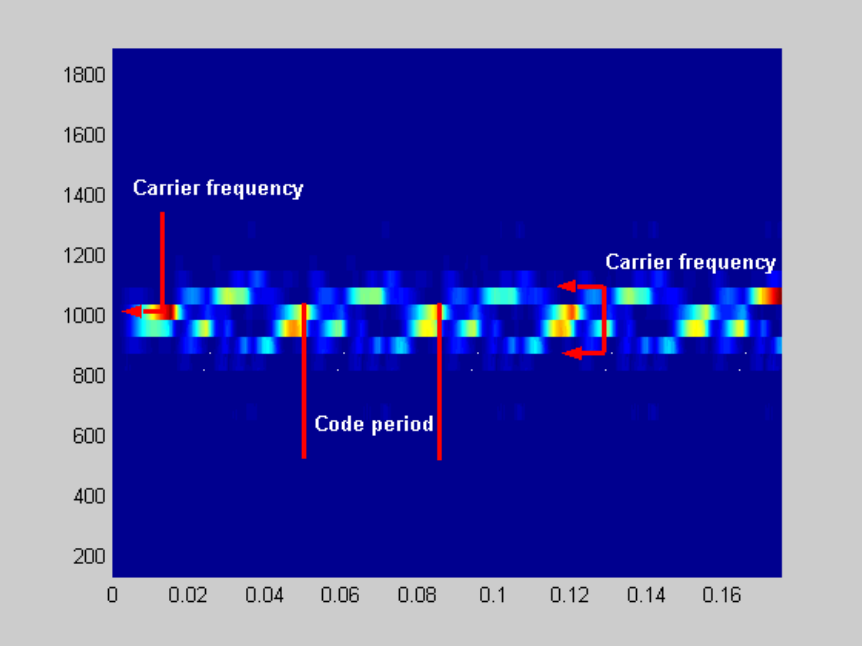


Figure 94 BPSK, 7-bit Barker code, 5 cycles per phase and signal only, zoom in the output after HOS.

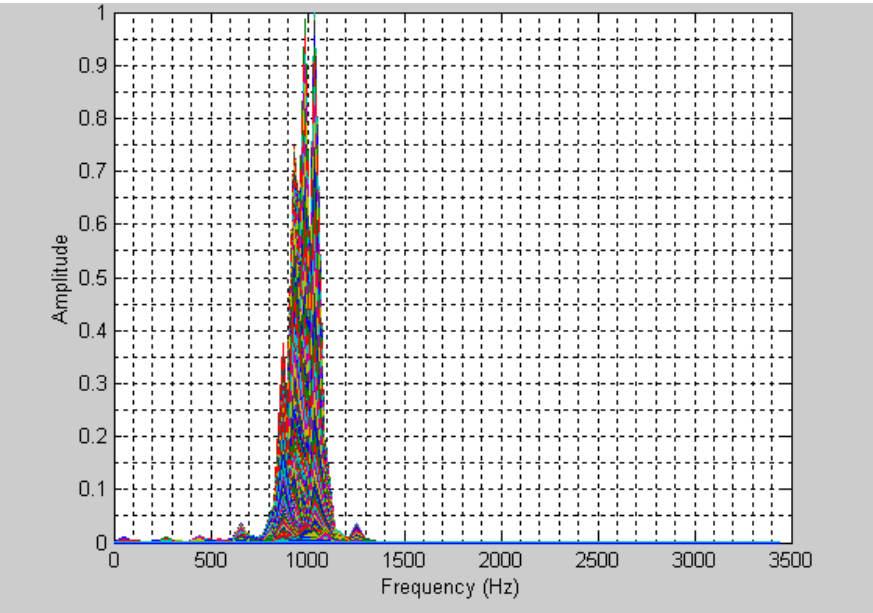


Figure 95 BPSK, 7-bit Barker code, 5 cycles per phase and signal only, amplitude-frequency plot .

2. BPSK, 7-bit Barker code, 5 cycles per phase and SNR=0 dB

Table 16 describes a BPSK signal with carrier frequency of 1 KHz, sampling frequency of 7 KHz, 7-bit Barker code, 5 cycles per bit and SNR = 0 dB. Code period and bandwidth are calculated by Equations (4.1.4) and (4.1.5).

BPSK - Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
Bandwidth (Hz)	200	218.64	
Bits per Barker code	7	-	
SNR (dB)	0	-	
Cycles per bit	5	-	
Code Period (ms)	35	35	

Table 16 BPSK, 7-bit Barker code, 5 cycles per phase and SNR=0 dB.

The modulation period of the BPSK signal is

$$t_c = \frac{(\text{Cycles/bit})(\text{Barker bits})}{f_c} = \frac{(5)(7)}{1000} = 35 \text{ ms} \tag{4.1.4}$$

The bandwidth of the signal depends on the cycles per bit (or chip) as

$$B = \frac{f_c}{\text{Cycles per bits}} = \frac{1000 \text{ Hz}}{5 \text{ cycles per bit}} = 200 \text{ Hz} \tag{4.1.5}$$

The PSD for the BPSK signal with SNR = 0 dB is shown in Figure 96. This plot provides a detailed description of the distribution of the power in the frequency domain, where the carrier frequency and the bandwidth of the signal can be compared from the results in (4.1.4) and (4.1.5).

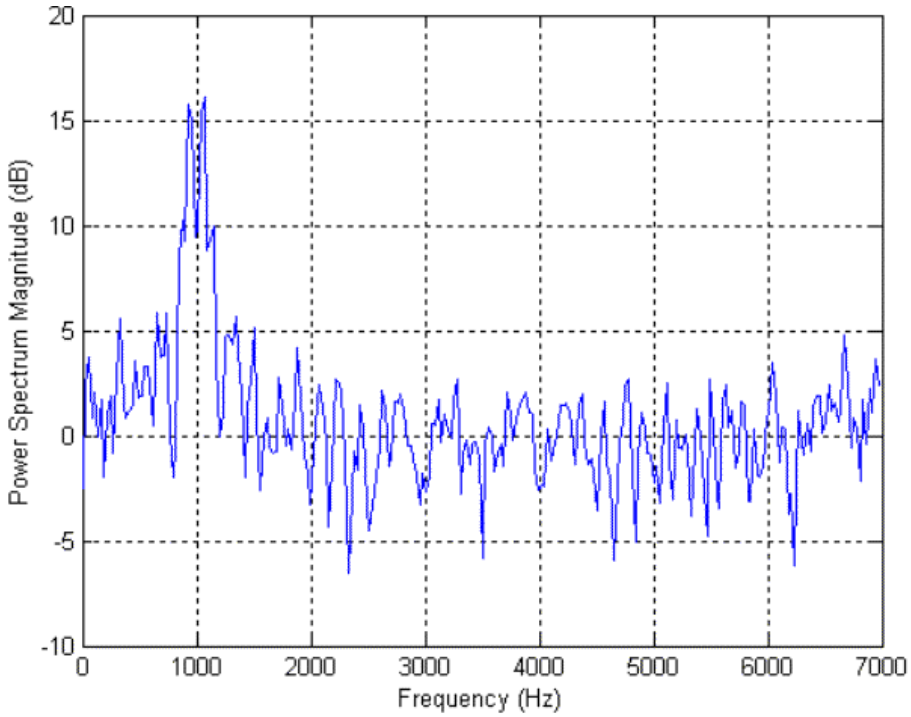
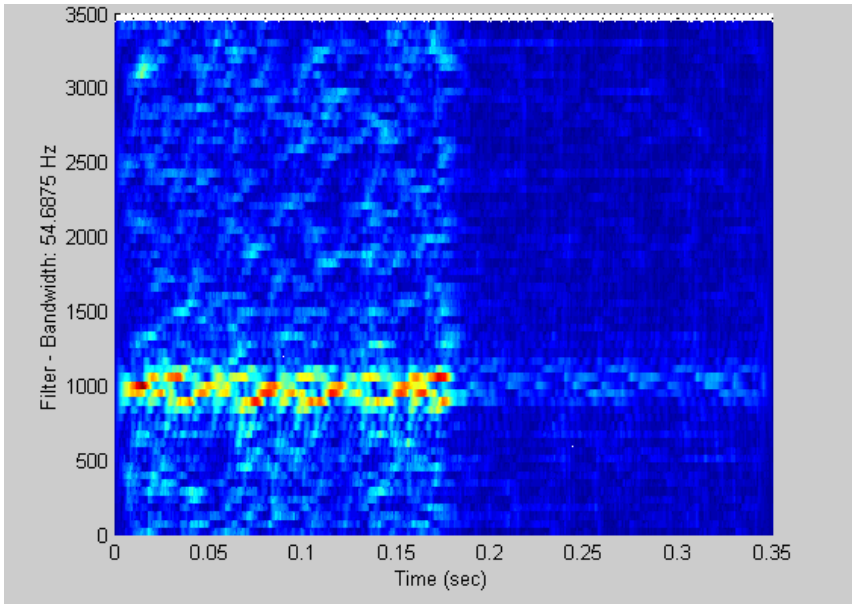


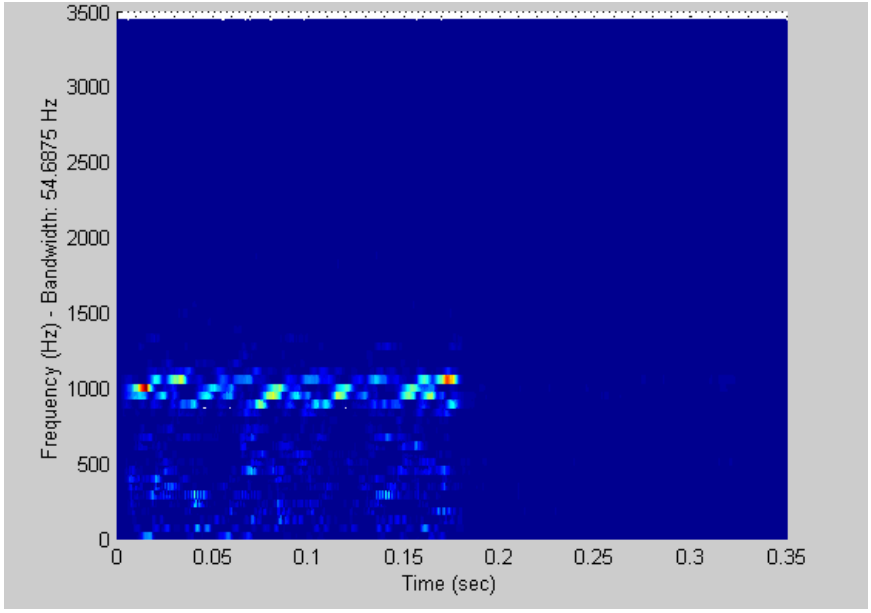
Figure 96 BPSK, 7-bit Barker code, 5 cycles per phase and SNR=0 dB PSD.

Figure 97 shows the resulting plots from the signal processing. Figure 97 (a) illustrates the output of the signal after the filter bank. Even though it provides a good description of the signal in the time-frequency domain, noise is presented along the signal. Figure 97 (b) proves the effectiveness of the third-order estimator to suppress the Gaussian noise. Most of the parameters can be identified and estimated.

Figure 98 shows a zoom in the resulting plot after HOS. Carrier frequency, bandwidth and modulation period are measured and recorded in the previous table. Figure 99 provides an amplitude-frequency plot.



(a)



(b)

Figure 97 BPSK, 7-bit Barker code, 5 cycles per phase and SNR=0 dB (a) Output of the parallel filter arrays (b) Output after HOS.

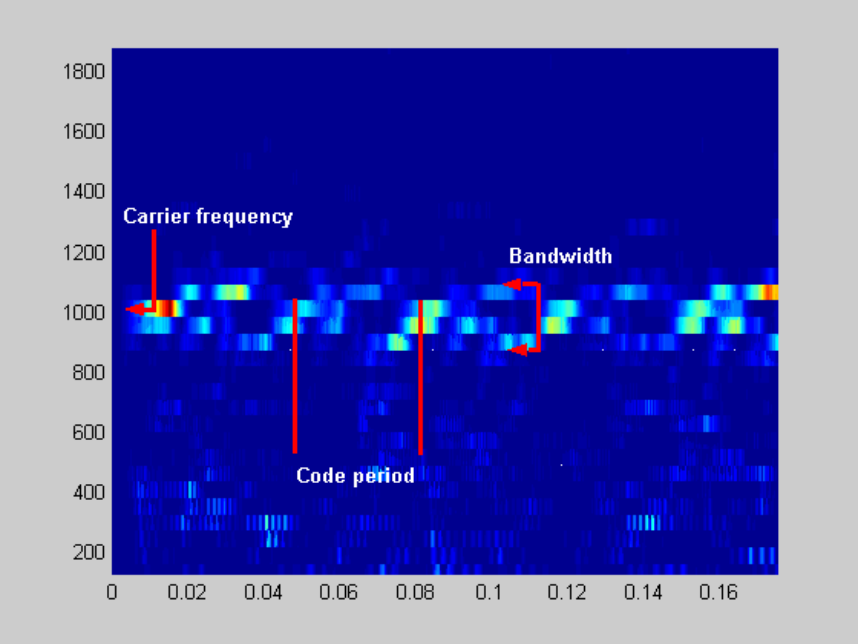


Figure 98 BPSK, 7-bit Barker code, 5 cycles per phase and SNR=0 dB, zoom in the output after HOS.

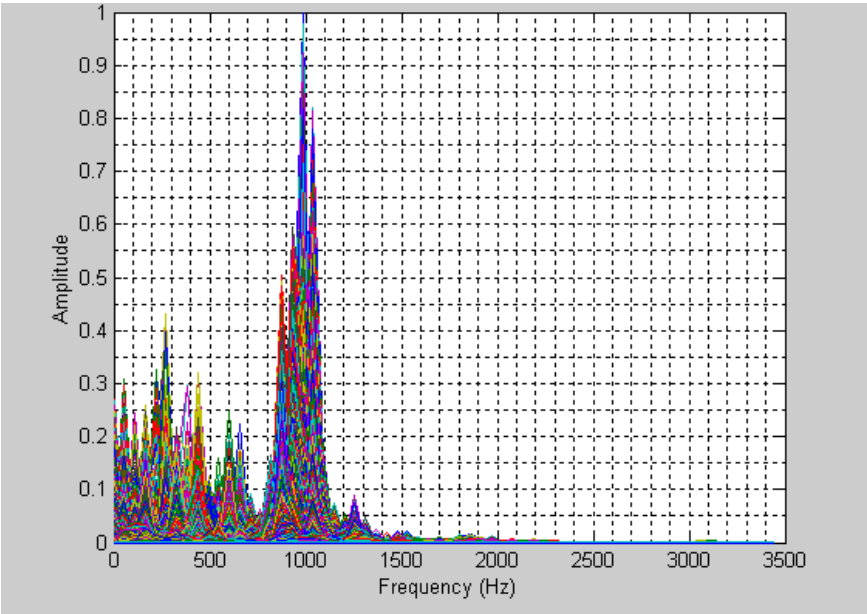


Figure 99 BPSK, 7-bit Barker code, 5 cycles per phase and SNR=0 dB, amplitude-frequency plot.

3. Summary

Figure 100 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth, code period and bits in Barker code in the BPSK signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very precise values for carrier frequency, bandwidth and code period except when the SNR is less than –6 dB. The number of bits in the Barker code can be extracted only when noise is not added to the signal.

BPSK	Carrier freq.(Hz)	Bandwidth (Hz)	Code period (ms)	Bits/code
Signal only	117%	117%	100%	100%
0 dB	122%	122%	100%	0%
(-) 6 dB	55%	117%	50%	0%

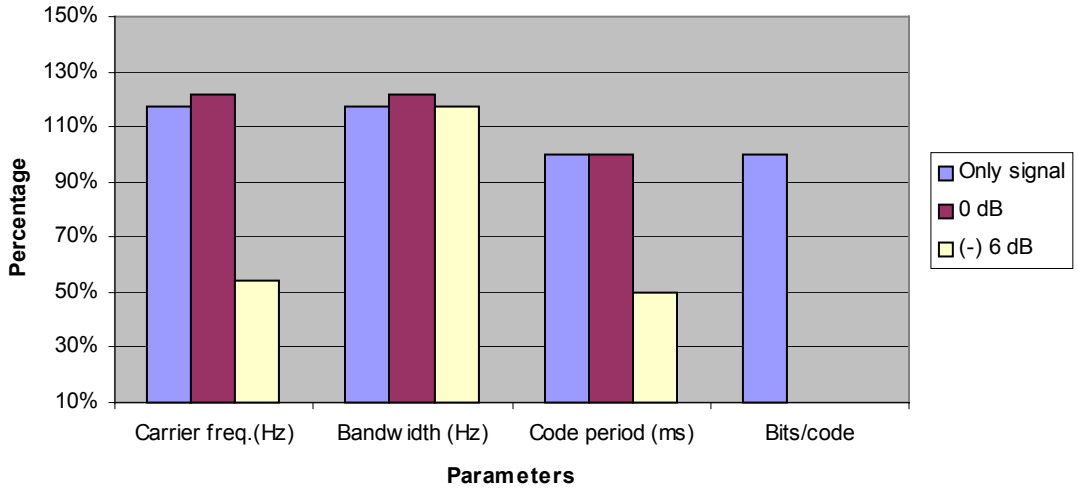


Figure 100 Performance of the signal processing detecting BPSK signals.

C. FMCW

The most popular modulation for LPI applications is the triangular modulation of a Frequency Modulated Continuous Wave (FMCW). As said before, the linear FMCW emitter uses a continuous 100 % duty-cycle waveform so that both the target range and the Doppler information can be measured unambiguously while maintaining a LPI [1]. The triangular modulation consists of two linear frequency modulation sections with positive and negative slopes. With this configuration, the range and Doppler frequency of the detected target can be extracted unambiguously by taking the sum and the difference of the two beat frequencies.

With the objective to demonstrate the efficiency of the proposed signal processing, twelve different signals are analyzed. The most important parameters have been varied, such as modulation bandwidth, modulation period and SNR. Carrier frequency and sampling frequency were kept constant due to detecting carrier frequency doesn't present any complexity. Table 18 shows the complete set of input signal to be analyzed. Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	File Name	Carrier Frequency	Sampling Frequency	Mod. Bandwidth	Mod. Period	SNR
1	F 1 7 250 20 s	1000	7000	250	20	-
2	F 1 7 250 20 0	1000	7000	250	20	0
3	F 1 7 250 20 -6	1000	7000	250	20	-6
4	F 1 7 250 30 s	1000	7000	250	30	-
5	F 1 7 250 30 0	1000	7000	250	30	0
6	F 1 7 250 30 -6	1000	7000	250	30	-6
7	F 1 7 500 20 s	1000	7000	500	20	-
8	F 1 7 500 20 0	1000	7000	500	20	0
9	F 1 7 500 30 -6	1000	7000	500	20	-6
10	F 1 7 500 30 s	1000	7000	500	30	-
11	F 1 7 500 30 0	1000	7000	500	30	0
12	F 1 7 500 30 -6	1000	7000	500	30	-6

Table 17 Matrix of input signals for FMCW.

1. FMCW ΔF=500 Hz t_m=20 ms signal only

Table 18 shows the input data and the results for a triangular FMCW signal with modulation bandwidth of 500 MHz. The objective of this example is to introduce changes in the bandwidth of the input signal to observe the responses of the proposed signal processing. Figure 101 illustrates the PSD of the input signal.

FMCW – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
Modulation Bandwidth (Hz)	500	492.12	
SNR (dB)	Signal only	-	
Number of triangles	5	-	
Modulation Period (ms)	20	20	

Table 18 FMCW ΔF=500 Hz t_m=20 ms signal only.

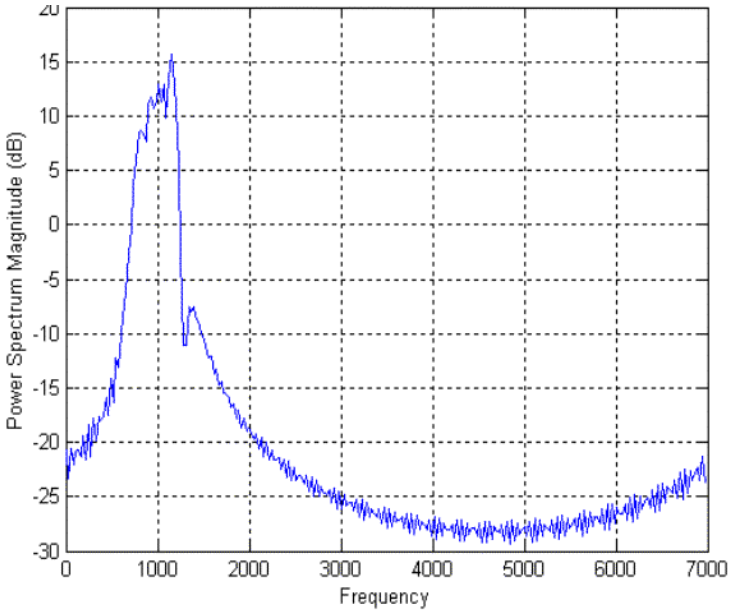
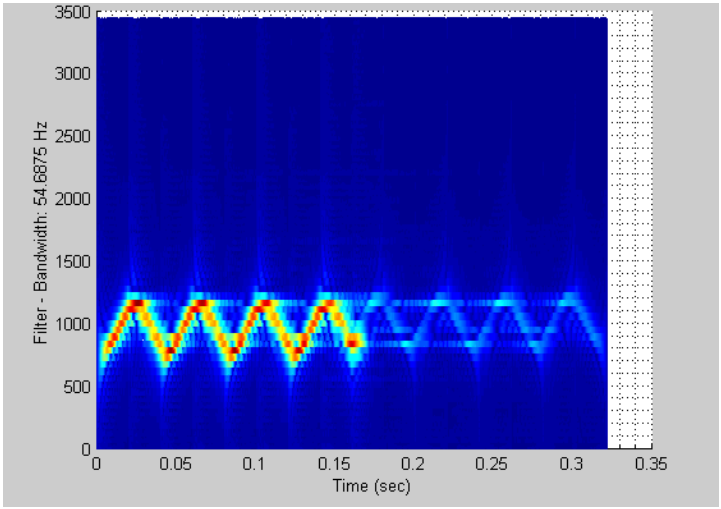
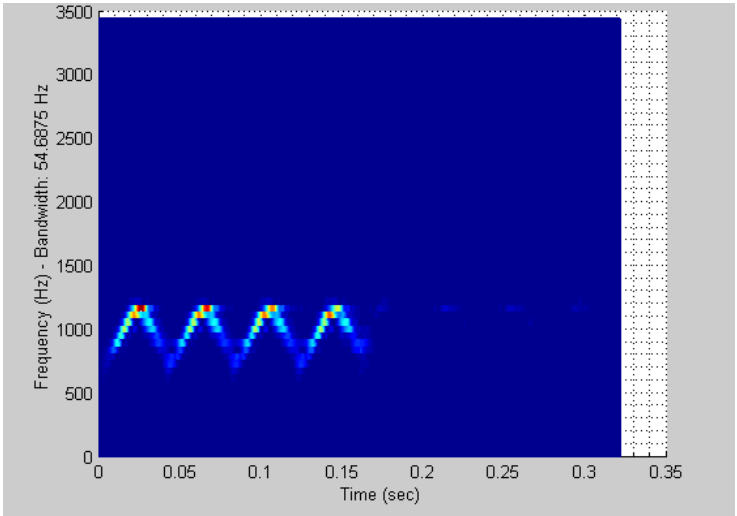


Figure 101 FMCW ΔF=500 Hz t_m=20 ms signal only PSD.

Figure 102 (a) represents the output of the parallel filter arrays. A perfect frequency-time representation of the signal can be observed alone many other characteristics. Figure 102 (b) shows the resulting plot after the third-order cumulant estimator. Although no noise is presented, other unwanted components of the signal are eliminated providing a clear representation of the signal. The observed modulation bandwidth is between 492.12 Hz and 546.8 Hz due to the error on the measurement is equal to the bandwidth of each sub-filter (54.68 Hz)



(a)



(b)

Figure 102 FMCW ΔF=500 Hz t_m=20 ms signal only (a) Output of the parallel filter arrays (b) Output after HOS.

Figure 103 shows a zoom in the resulting plot after HOS. Some parameters are estimated such as carrier frequency equal to 1 KHz, modulation bandwidth equal to 500 Hz and modulation period of 20 ms. Figure 104 Presents an amplitude-frequency view of the resulting plot after HOS, A concentration of frequency components around the carrier frequency and the bandwidth occupied by the signal can be seen.

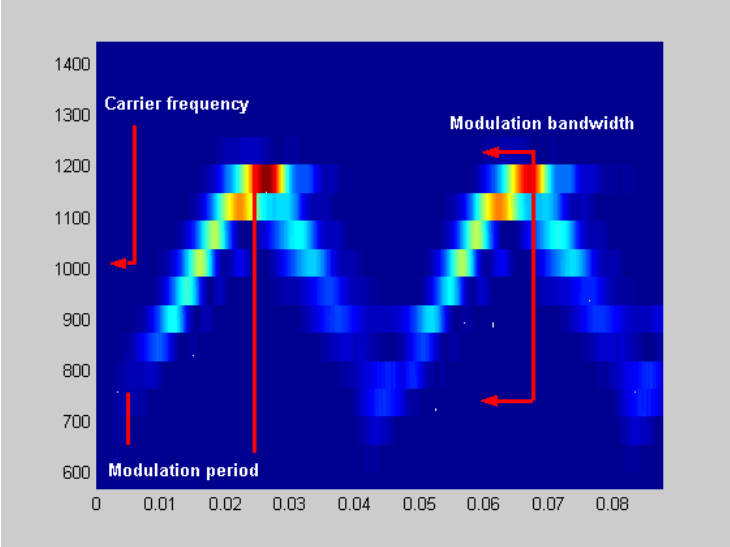


Figure 103 FMCW ΔF=500 Hz t_m=20 ms signal only zoom in output of HOS.

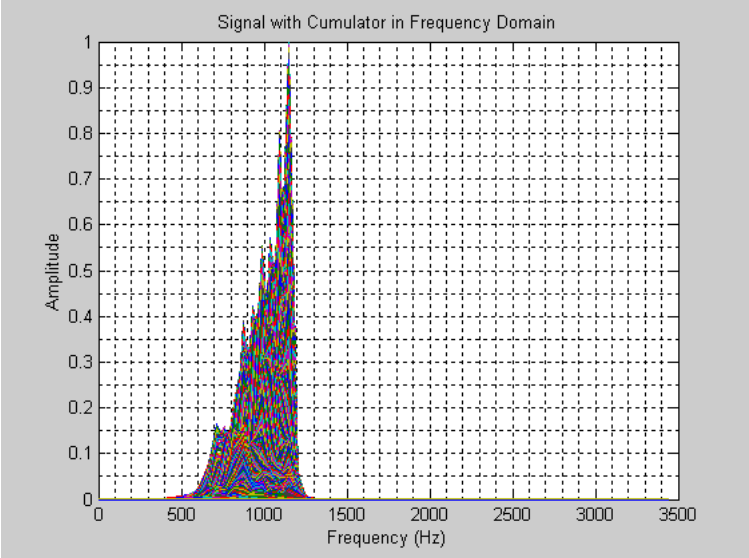


Figure 104 FMCW ΔF=500 Hz t_m=20 ms signal only amplitude-frequency plot.

2. FMCW ΔF=500 Hz t_m=20 ms SNR= 0 dB

Table 19 shows the input and obtained data for a triangular FMCW signal with SNR= 0 dB. Figure 105 illustrates the PSD of the signal centered in a carrier frequency of 1 KHz.

FMCW - Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
Modulation Bandwidth (Hz)	500	492.12	
SNR (dB)	0	-	
Number of triangles	5	-	
Modulation Period (ms)	20	20	

Table 19 FMCW ΔF=500 Hz t_m=20 ms SNR= 0 dB.

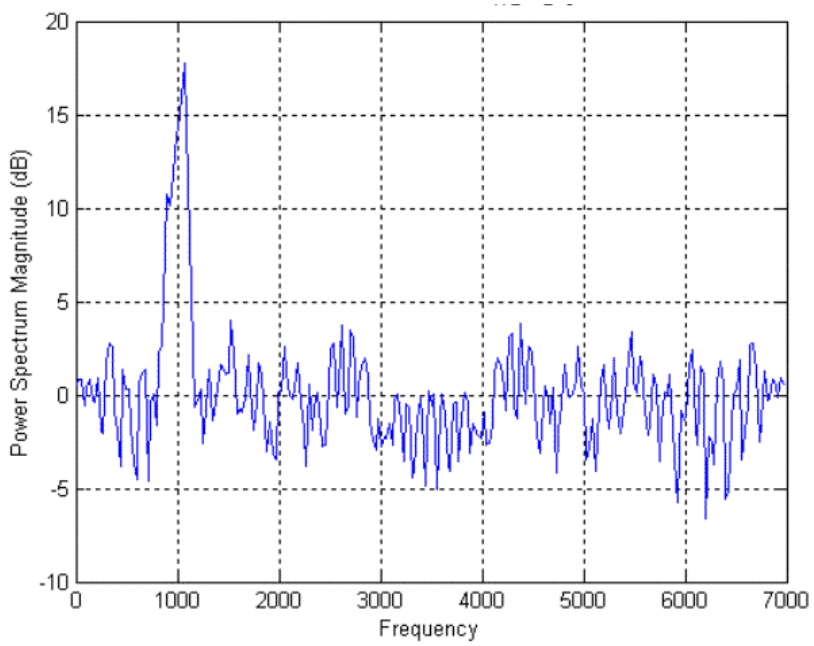
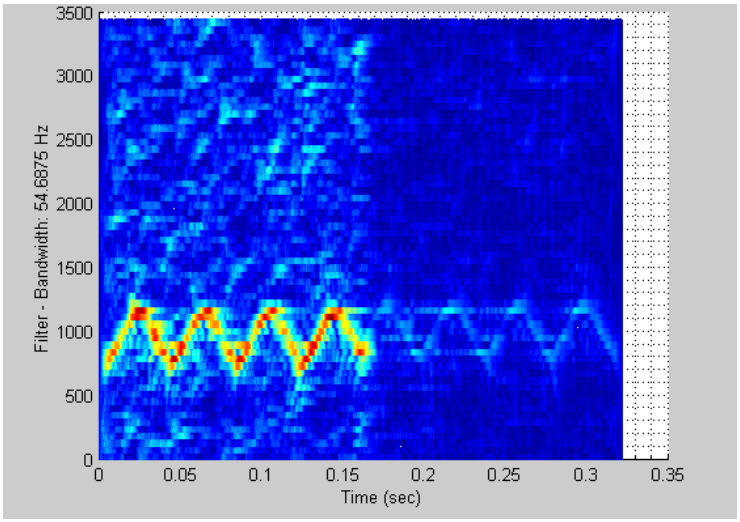
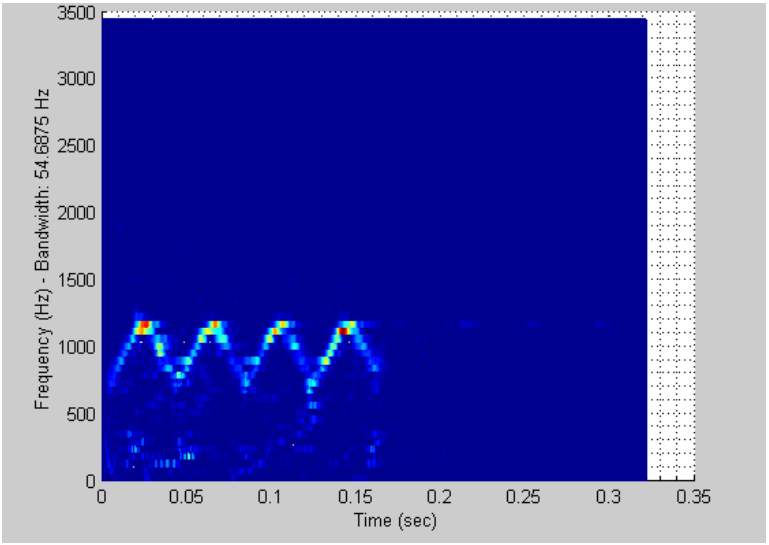


Figure 105 FMCW ΔF=500 Hz t_m=20 ms SNR= 0 dB PSD.

Figure 106 (a) shows the output of the filter bank. Although a same level of noise and signal are presented, the plot presents a good frequency-time description of the signal. Figure 106 (b) shows the resulting plot after the third-order estimators. Most of the Gaussian noise has been eliminated and some inter-modulation products starts appearing at low-frequency filters.



(a)



(b)

Figure 106 FMCW ΔF=500 Hz t_m=20 ms SNR= 0 dB (a) Output of the parallel filter arrays (b) Output after HOS.

Figure 107 shows a zoom in the resulting plot after applying HOS. Most of the parameters identify the signal can be measured. Figure 108 presents an amplitude-frequency view of the resulting plot.

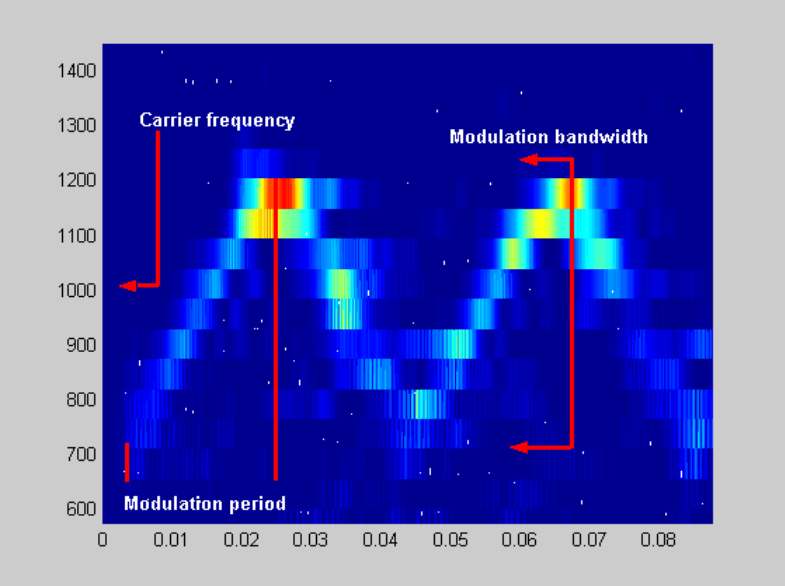


Figure 107 FMCW ΔF=500 Hz t_m=20 ms SNR= 0 dB zoom in the resulting plot after HOS.

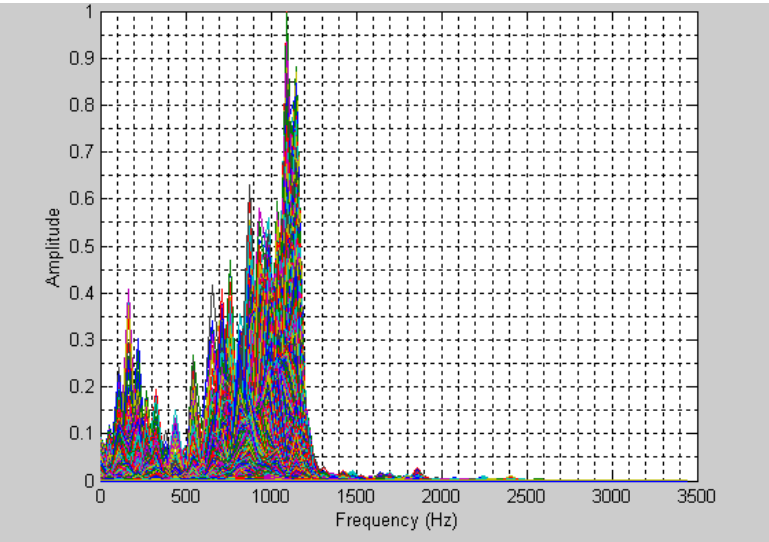


Figure 108 FMCW ΔF=500 Hz t_m=20 ms SNR= 0 dB amplitude-frequency plot.

3. Summary

Figure 109 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, modulation bandwidth and modulation period in the FMCW signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very accurate values for carrier frequency, modulation bandwidth and modulation period in any environment analyzed. The performance of the proposed signal processing has demonstrated being very efficient to detect triangular FMCW radar signals.

FMCW	Carrier freq.(Hz)	Mod.Bandwidth (Hz)	Mod.Period(ms)
Signal only	100%	100%	100%
0 dB	100%	100%	100%
(-) 6 dB	100%	94%	100%

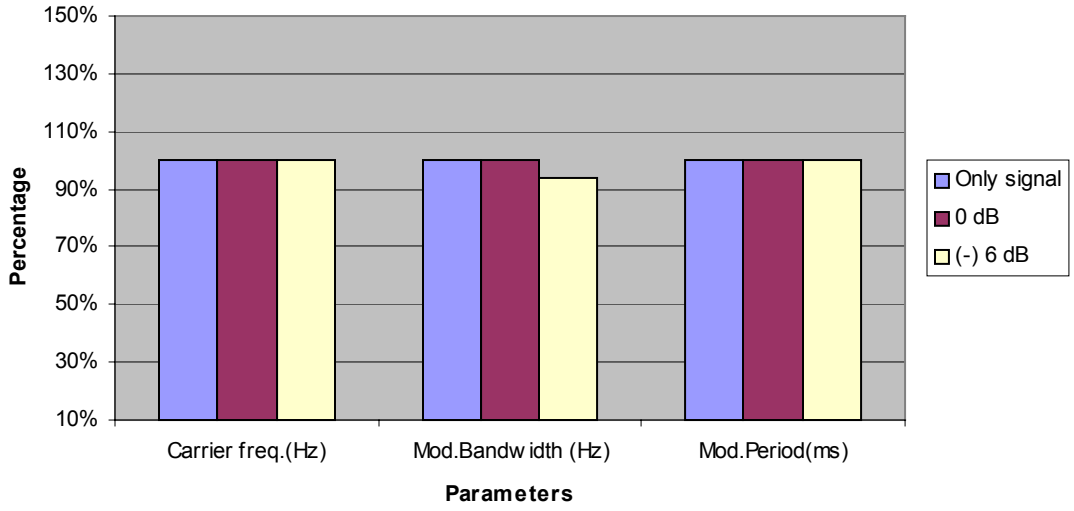


Figure 109 Performance of the signal processing detecting FMCW signals.

D. FRANK POLYPHASE CODE

Frank codes belong to the family of polyphase codes, Chirp codes and Barker codes, and have been successfully used in implementing LPI radar signals. A Frank-coded waveform consists of a constant amplitude signal whose carrier frequency is modulated by the phases of the Frank code. Each element of the code is τ seconds along, which is approximately equal to the reciprocal of the waveform 3-dB bandwidth. A Frank code has a length of N^2, which is the result obtained from the multiplication of N frequency steps and N samples per frequency.

To test the performance of the proposed signal processing, twelve different Frank-coded signals were generated. Number of phases, code period and SNR were varied to cover different variations of the signal without noise and in very noisy environments as shown in Table 20 . This thesis presents the analysis of one test signal. Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	File	Carrier Frequency	Sampling Frequency	N(Phases)	Cycles/phase	SNR
1	FR_1_7_4_1_s	1000	7000	4	1	-
2	FR_1_7_4_1_0	1000	7000	4	1	0
3	FR_1_7_4_1_-6	1000	7000	4	1	-6
4	FR_1_7_4_5_s	1000	7000	4	5	-
5	FR_1_7_4_5_0	1000	7000	4	5	0
6	FR_1_7_4_5_-6	1000	7000	4	5	-6
7	FR_1_7_8_1_s	1000	7000	8	1	-
8	FR_1_7_8_1_0	1000	7000	8	1	0
9	FR_1_7_8_1_-6	1000	7000	8	1	-6
10	FR_1_7_8_5_s	1000	7000	8	5	-
11	FR_1_7_8_5_0	1000	7000	8	5	0
12	FR_1_7_8_5_-6	1000	7000	8	5	-6

Table 20 Matrix of input signals for Frank Polyphase Code.

1. Frank Code, N=16, cycles per phase =5 and signal only

Table 21 describes a Frank-coded signal with carrier frequency equal to 1 KHz, sampling frequency equal to 7 KHz, 16 phases and 5 cycle per phase. Code Period and Bandwidth are calculated in Equations (4.1.6) and (4.1.7). In addition, Table 21 shows the results obtained by processing the signal with a parallel filter arrays and HOS. Figure 110 illustrates the PSD of the input signal.

Frank - Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N - Phases	16		
SNR (dB)	Signal only	-	
Cycles per phase	5	-	
Bandwidth	200	218.72	
Code Period (ms)	80	80	

Table 21 Frank Code N=16 cycles per phase =5 signal only.

The code period of the Frank-coded signal is

$$t_c = \frac{(\text{Cycles/phase})(\text{N}^2)}{f_c} = \frac{(5)(16)}{1000} = 80 \text{ ms} \tag{4.1.6}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{5 \text{ cycles per phase}} = 200 \text{ Hz} \tag{4.1.7}$$

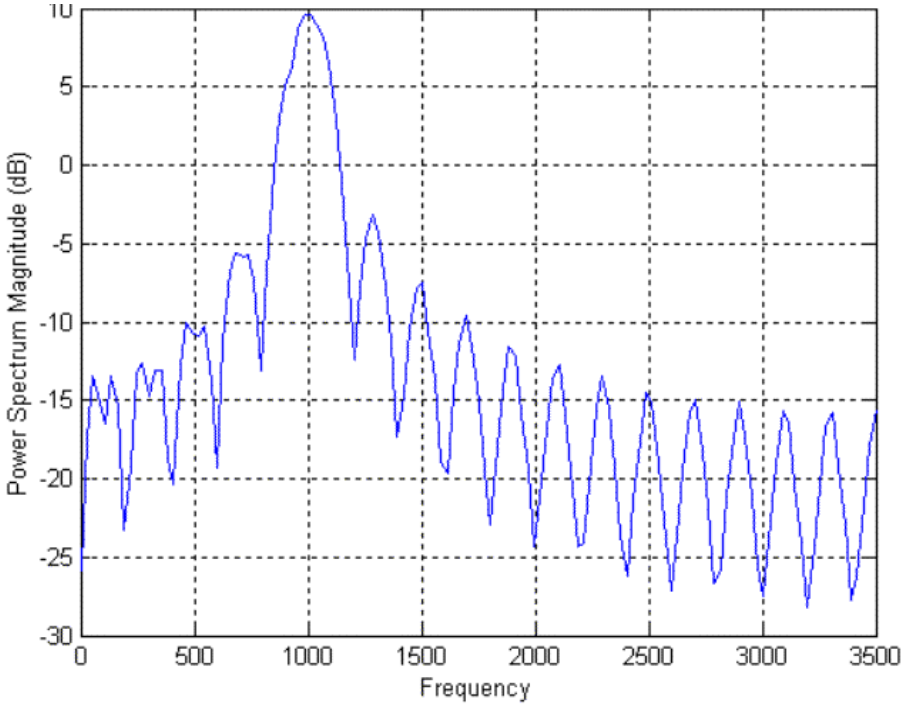
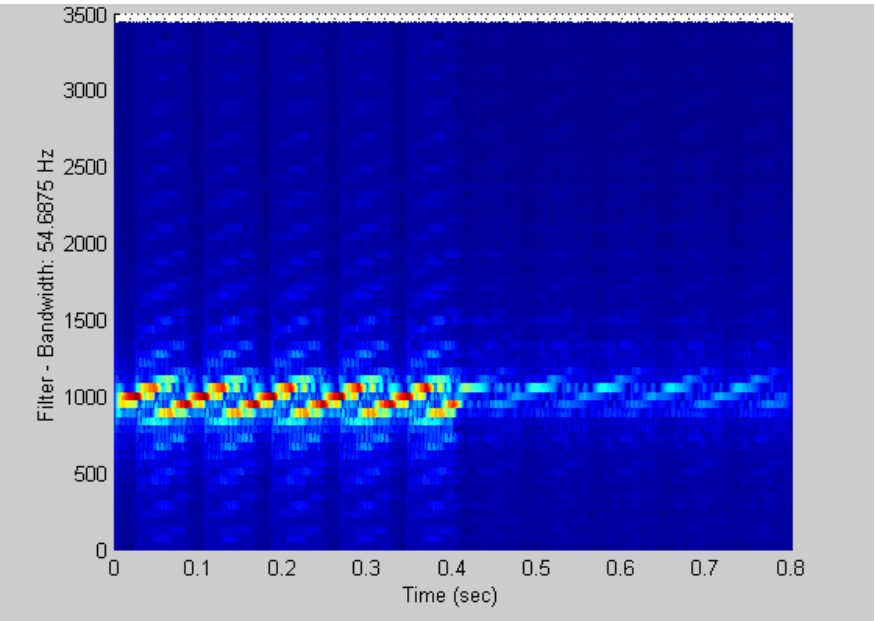


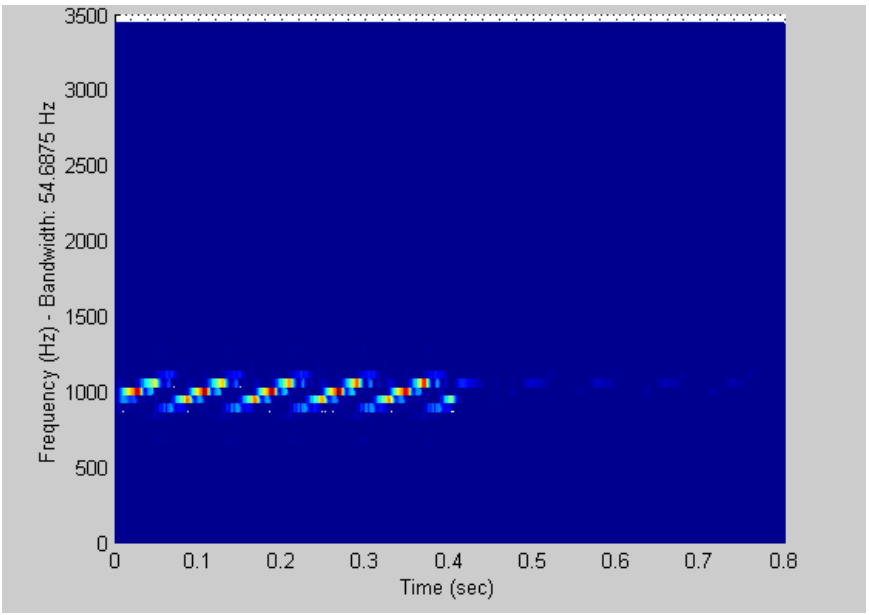
Figure 110 Frank Code N=16 cycles per phase =5 signal only PSD

Figure 111 (a) presents the response of the parallel filter arrays after processing the input signal. This signal is divided in smaller frequencies, to show a complete frequency-time description of the signal. As said before, the bandwidth of each sub-filter is 54.68 Hz as a result of using 64 filters and sampling frequency equal to 7000 Hz. Figure 111 (b) shows the resulting plot after the third-order cumulant estimators are applied.

Figure 112 (a) makes a zoom in the resulting plot after HOS. This plot provides information about the carrier frequency, bandwidth and code period. All this data is recorded and compared with the input data in Table 21 . Figure 112 (b) shows the phase shift of the signal.

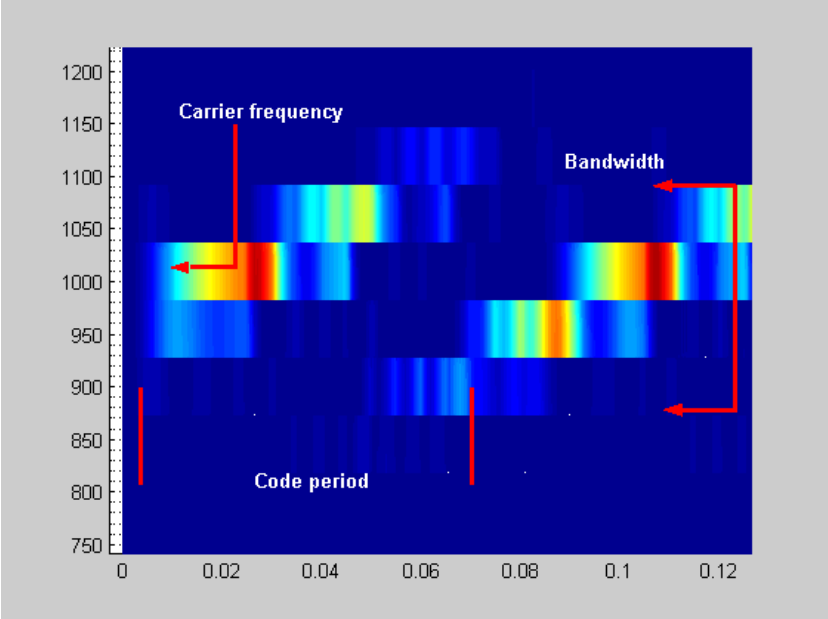


(a)

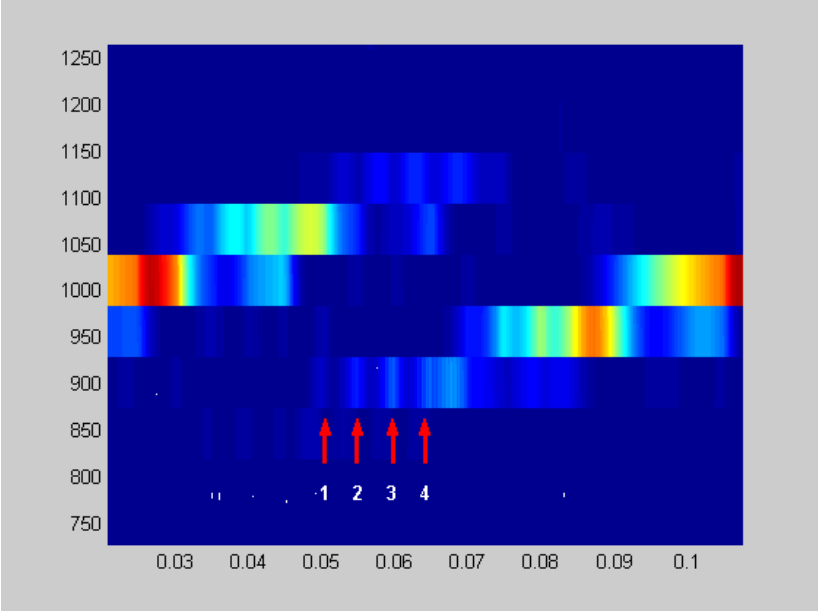


(b)

Figure 111 Frank Code N=16 cycles per phase =5 signal only (a) Output of the parallel filter arrays (b) Output after HOS.

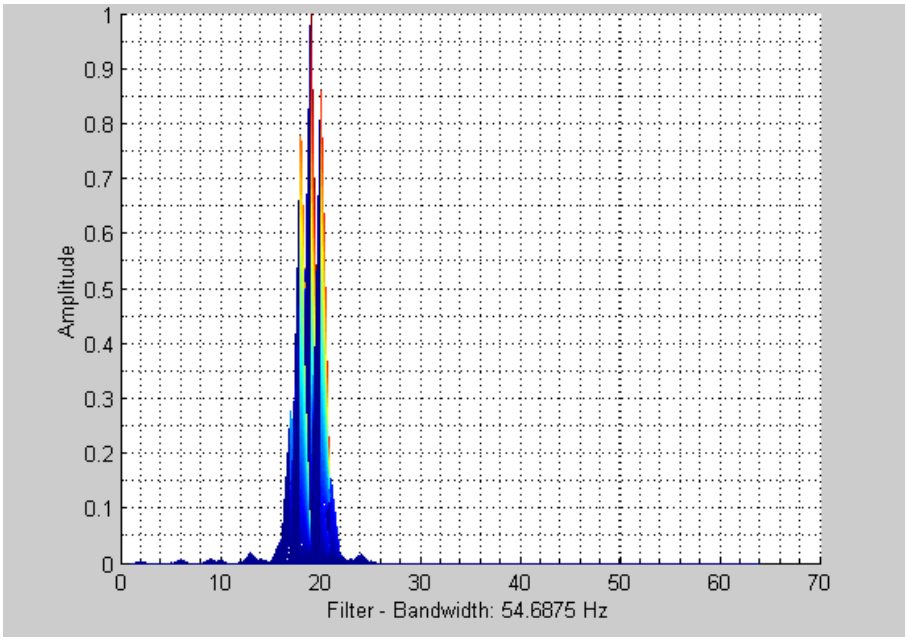


(a)

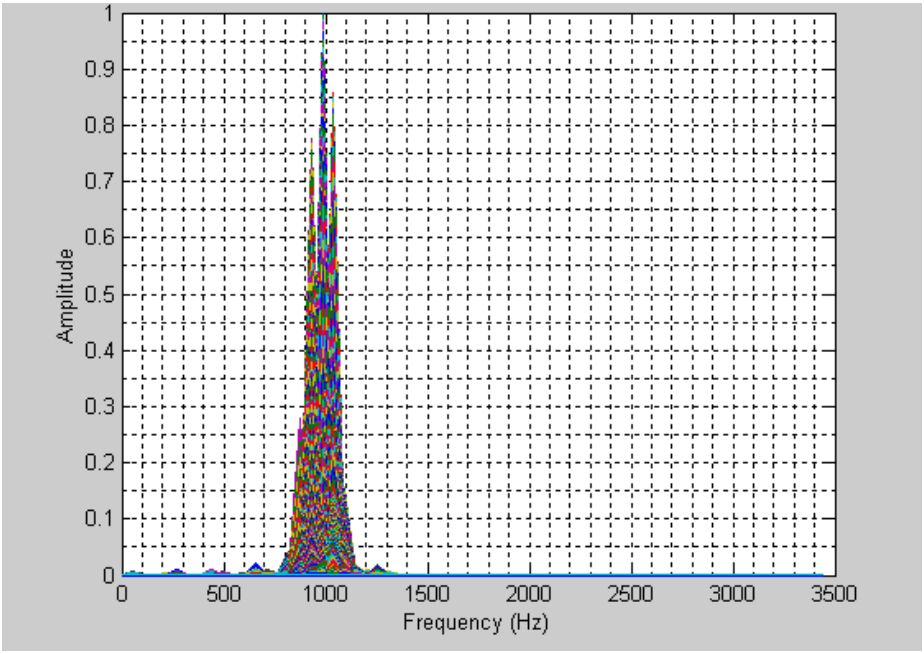


(b)

Figure 112 Frank Code N=16 cycles per phase =5 signal only (a) Zoom in the resulting plot after HOS (b) Phase shift.



(a)



(b)

Figure 113 Frank Code N=16 cycles per phase =5 signal only (a) Amplitude-filter plot (b) Amplitude-frequency plot.

.

2. Frank Code, N=16, cycles per phase =5 and SNR=0 dB

Table 22 describes a Frank-coded signal with carrier frequency equal to 1 KHz, sampling frequency equal to 7 KHz, 16 phases, 5 cycle per phase and SNR = 0 dB. Code Period and Bandwidth are calculated in Equations (4.1.8) and (4.1.9). In addition, Table 22 shows the results obtained by processing the signal with a parallel filter arrays and HOS. Figure 114 illustrates the PSD of the input signal.

Frank - Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N - Phases	16		
SNR (dB)	0	-	
Cycles per phase	5	-	
Bandwidth	200	218.72	
Code Period (ms)	80	80	

Table 22 Frank Code N=16 cycles per phase =5 SNR=0 dB.

The code period of the Frank-coded signal is

$$t_c = \frac{(\text{Cycles/phase})(\text{N}^2)}{f_c} = \frac{(5)(16)}{1000} = 80 \text{ ms} \tag{4.1.8}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{5 \text{ cycles per phase}} = 200 \text{ Hz} \tag{4.1.9}$$

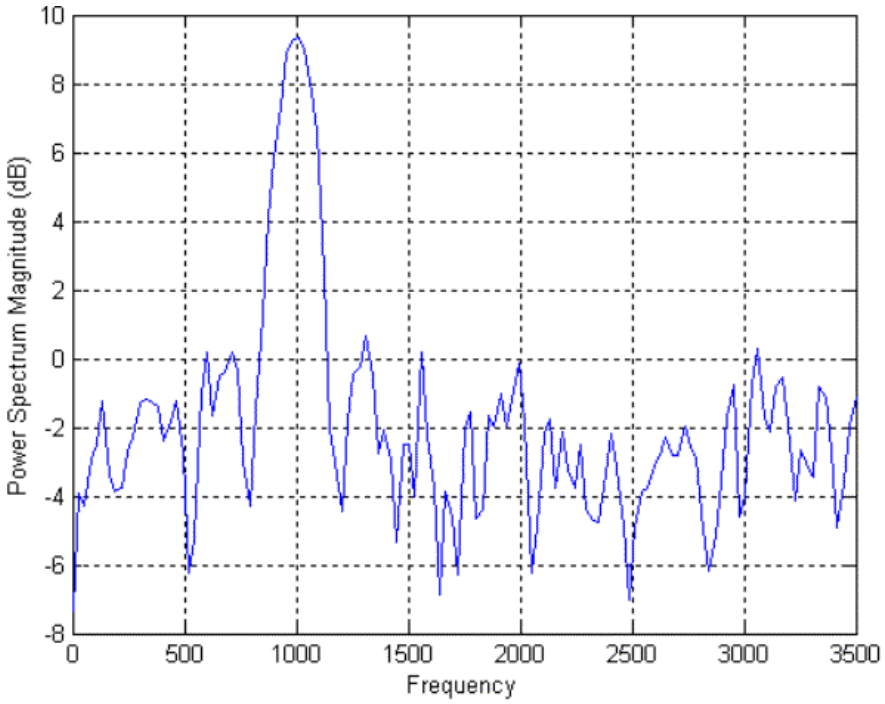
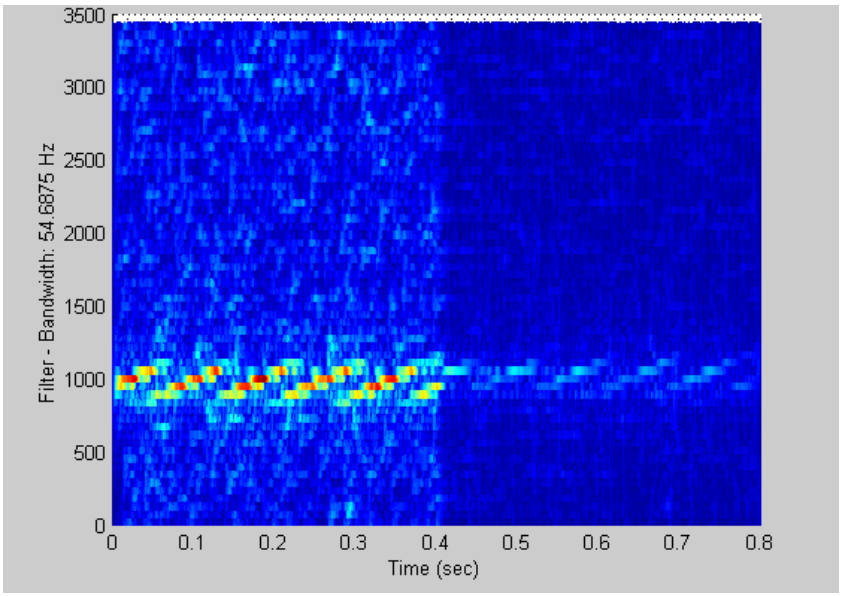


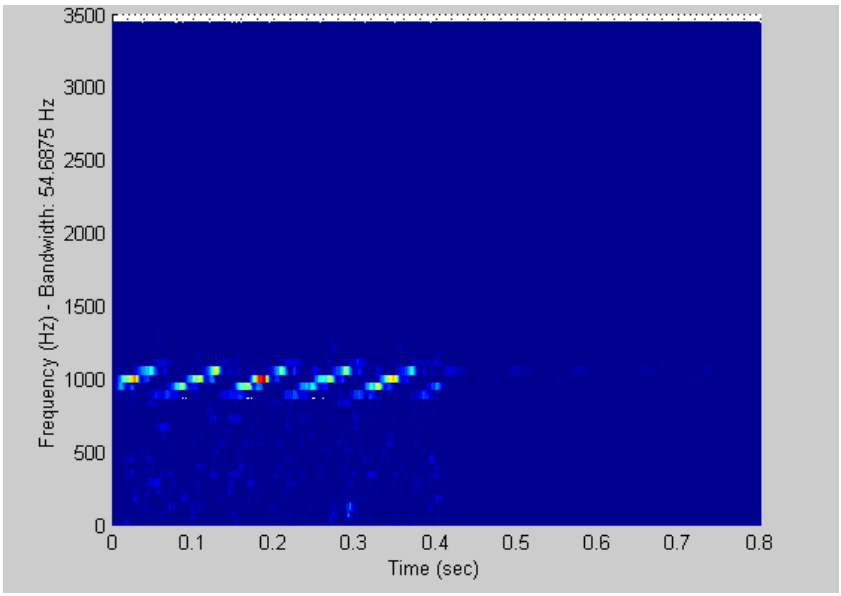
Figure 114 Frank Code N=16 cycles per phase =5 SNR=0 dB PSD.

Figure 115 (a) presents the response of the parallel filter arrays after processing the input signal. This signal is divided in smaller frequencies, to show a complete frequency-time description of the signal. Figure 115 (b) shows the resulting plot after the third-order cumulant estimators are applied. Because the SNR in the input signal is 0 dB, this plot shows how the noise is eliminated, facilitating the extraction of the needed parameters to identified the signal.

Figure 116 (a) makes a zoom in the resulting plot after HOS. This plot provides information about the carrier frequency, bandwidth and code period. All this data is recorded and compared with the input data in Table 22 . Figure 116 (b) shows the phase shift of the signal.

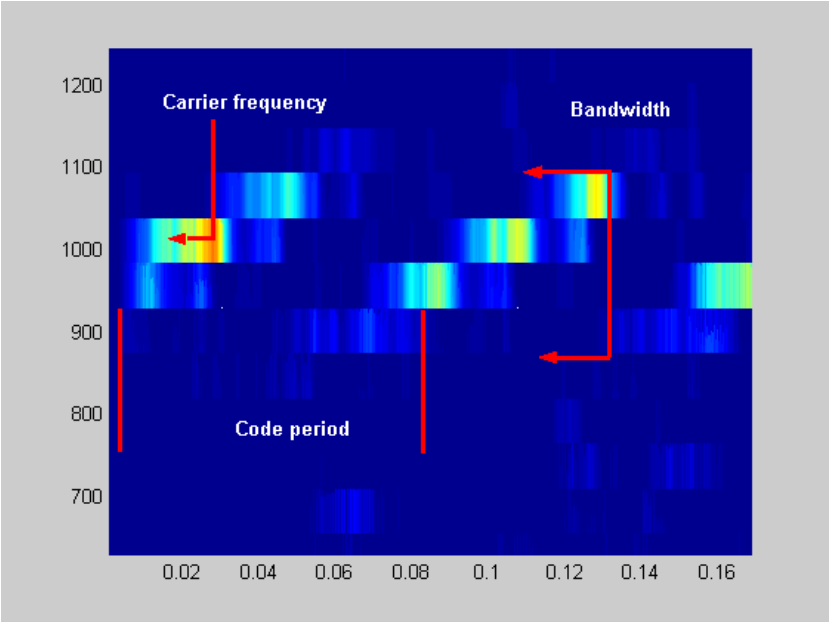


(a)

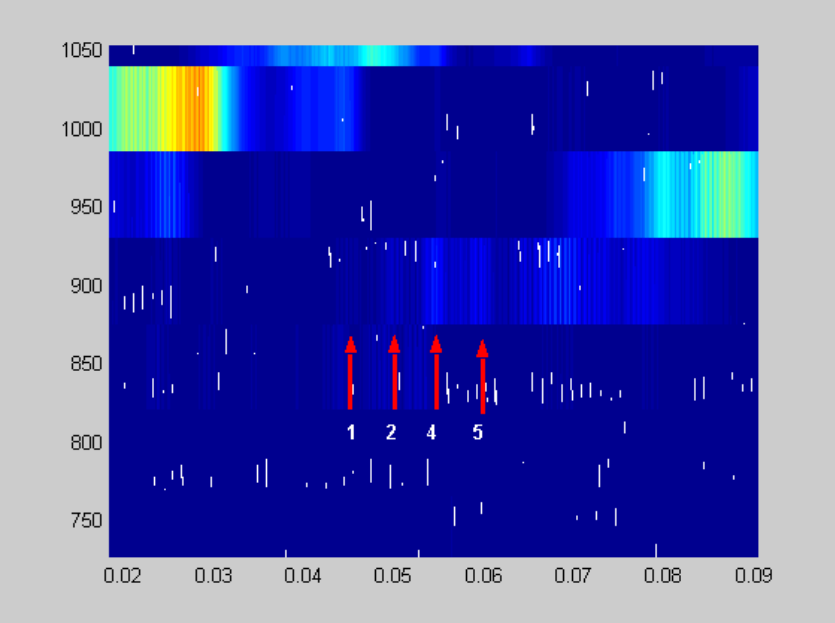


(b)

Figure 115 Frank Code N=16 cycles per phase =5 SNR=0 dB (a) Output of the parallel filter arrays (b) Output after HOS.

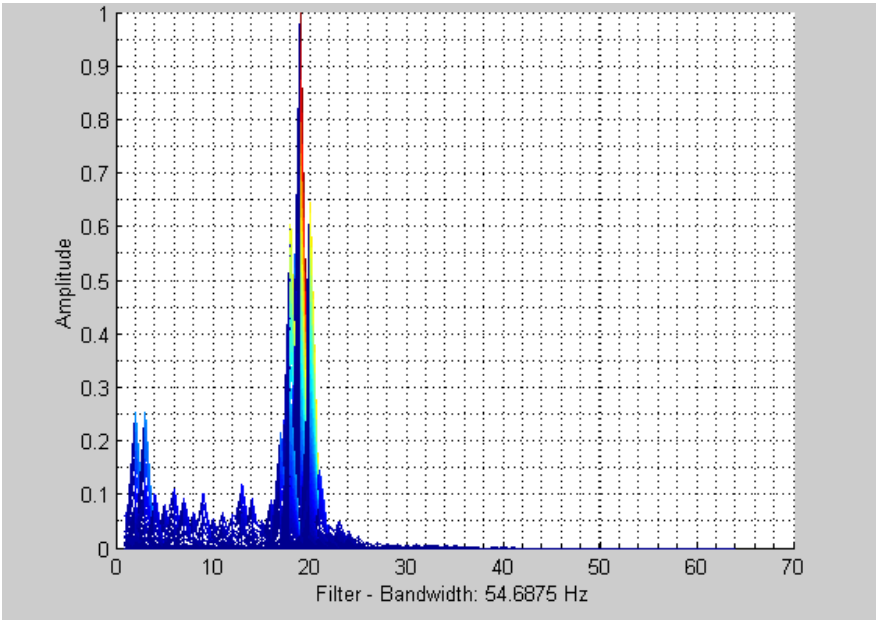


(a)

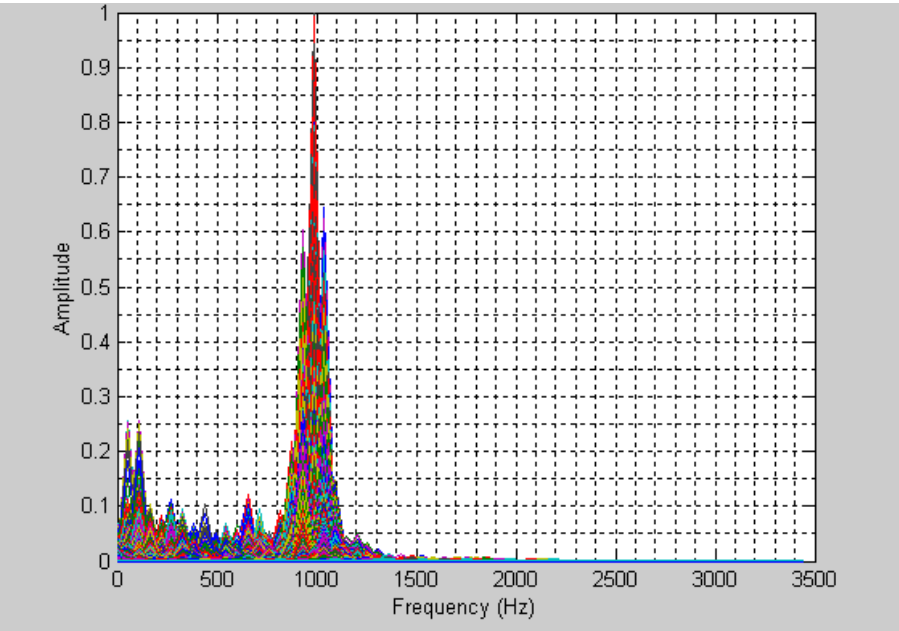


(b)

Figure 116 Frank Code N=16 cycles per phase =5 SNR=0 dB (a) Zoom in resulting plot after HOS (b) Phase shift: first 4 phases.



(a)



(b)

Figure 117 Frank Code N=16 cycles per phase =5 SNR=0 dB (a) Amplitude-filter plot (b) Amplitude-frequency plot.

3. Summary

Figure 118 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth, code period and number of phases in Frank-coded signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very accurate values for carrier frequency, bandwidth and code period in any environment analyzed with a low performance when SNR is less than –6 dB. The number of phases can be only extracted when noise is not presented and a number of cycles per phase is greater than 5.

FRANK CODE	Carrier freq.(Hz)	Bandwidth (Hz)	Code period (ms)	Phases
Signal only	100%	105%	100%	100%
0 dB	100%	103%	100%	0%
(-) 6 dB	100%	52%	72%	0%

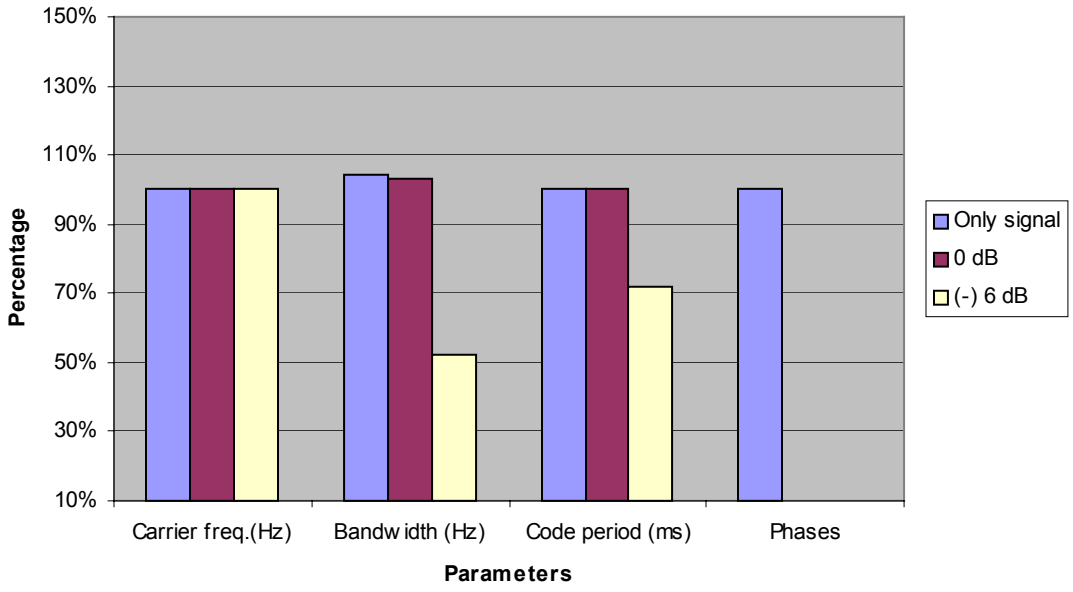


Figure 118 Performance of the signal processing detecting Frank-coded signals.

E. P1 POLYPHASE CODE

The detection and classification of polyphased-coded signals is especially efficient with the use of parallel filter arrays and HOS. With the object of demonstrating this fact, a set of twelve different P1-coded signals was generated (Table 23).

Three parameters were varied: number of phases, cycles per phase and SNR. As mention in Chapter II, P1-coded signal N=4 produces a matrix of 16 different phases. In the same way, N=8 produces a matrix of 64 phases.

In addition to the standard parameters previously provided, in the case of polyphased-coded signals the number of phases can be detected. This is a very important factor when performing recognition of a unknown signal. Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	File	Carrier Frequency	Sampling Frequency	N(Phases)	Cycles/phase	SNR
1	P1_1_7_4_1_s	1000	7000	4	1	-
2	P1_1_7_4_1_0	1000	7000	4	1	0
3	P1_1_7_4_1_-6	1000	7000	4	1	-6
4	P1_1_7_4_5_s	1000	7000	4	5	-
5	P1_1_7_4_5_0	1000	7000	4	5	0
6	P1_1_7_4_5_-6	1000	7000	4	5	-6
7	P1_1_7_8_1_s	1000	7000	8	1	-
8	P1_1_7_8_1_0	1000	7000	8	1	0
9	P1_1_7_8_1_-6	1000	7000	8	1	-6
10	P1_1_7_8_5_s	1000	7000	8	5	-
11	P1_1_7_8_5_0	1000	7000	8	5	0
12	P1_1_7_8_5_-6	1000	7000	8	5	-6

Table 23 Matrix of input signals for P1.

1. P1 Code N=64 cycles per phase =5 signal only

Table 24 shows the input values and the obtained parameters for a P1-coded signal. The calculation of the input values of code period and bandwidth is given by Equations (4.1.10) and (4.1.11). Figure 119 presents the PSD of the signal.

P1 - Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N – Phases	64	64	
SNR (dB)	Signal only	-	
Cycles per phase	5	-	
Bandwidth	200	218	
Code Period (ms)	320	320	

Table 24 P1 Code N =64 cycles per phase =5 signal only.

The code period of the P1 signal is

$$t_c = \frac{(\text{Cycles/bit})(N)}{f_c} = \frac{(5)(64)}{1000} = 320 \text{ ms} \tag{4.1.10}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{5 \text{ cycles per phase}} = 200 \text{ Hz} \tag{4.1.11}$$

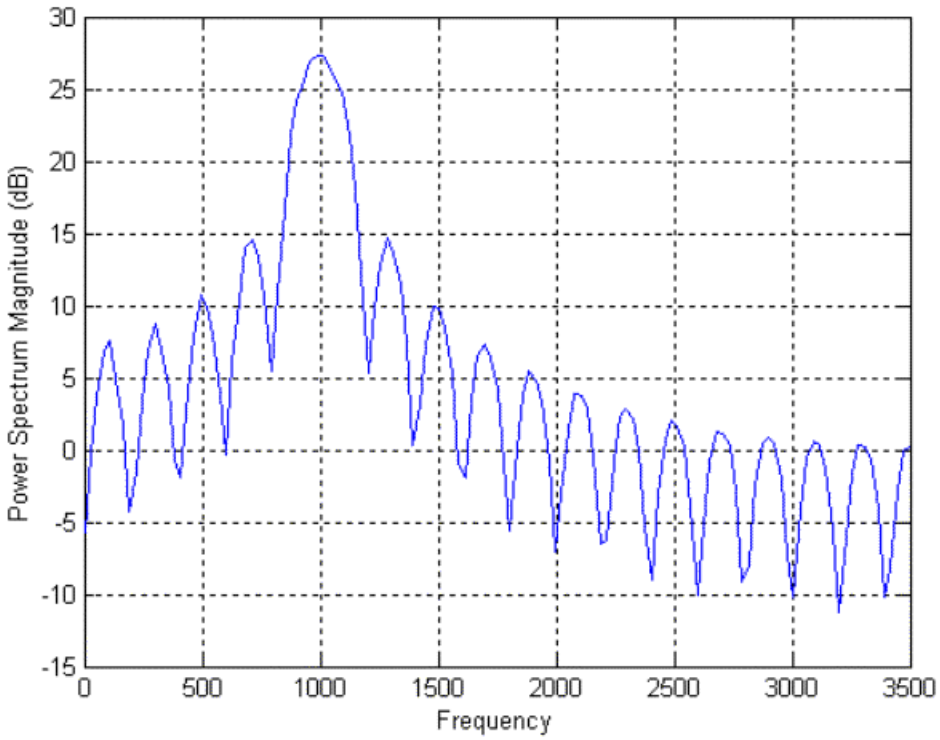
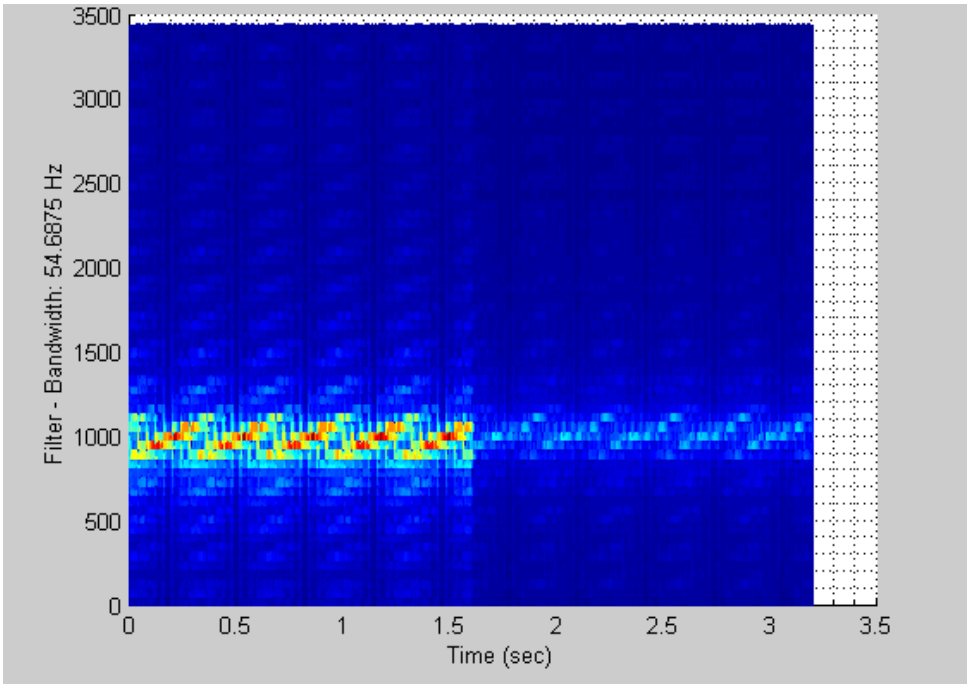
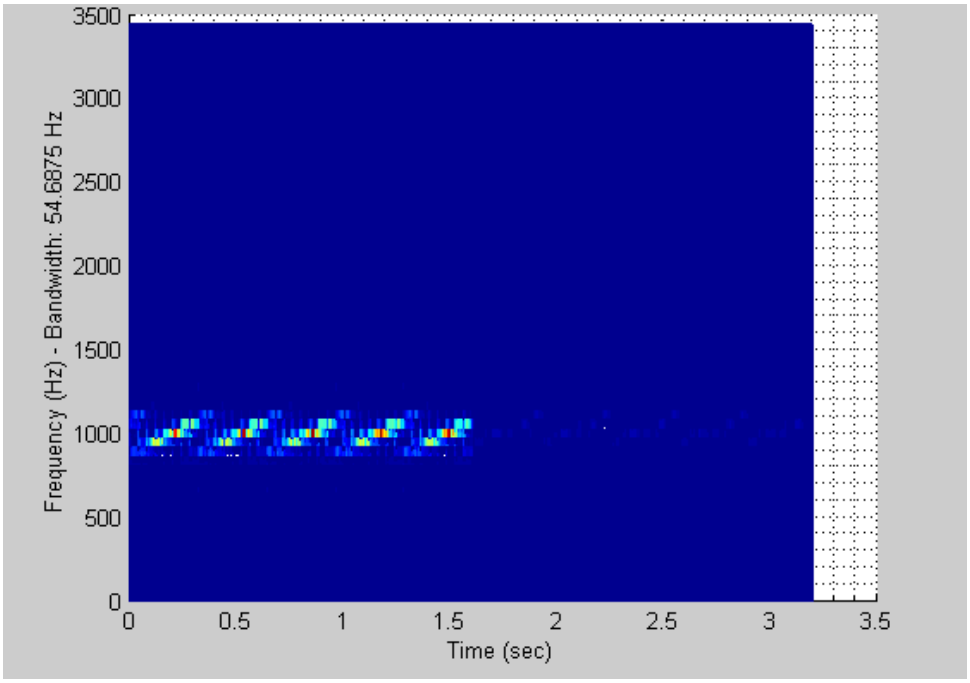


Figure 119 P1 Code *N*=64 cycles per phase =5 signal only PSD.

The output of the filter bank is shown in Figure 120 (a). This figure presents a frequency-time description of the signal where can be identified the most important parameters. Figure 120 (b) shows the output after the third-order estimators are applied to each one of sub-filters. Although there is not no noise added to the signal, it facilitates the estimation of the parameters.



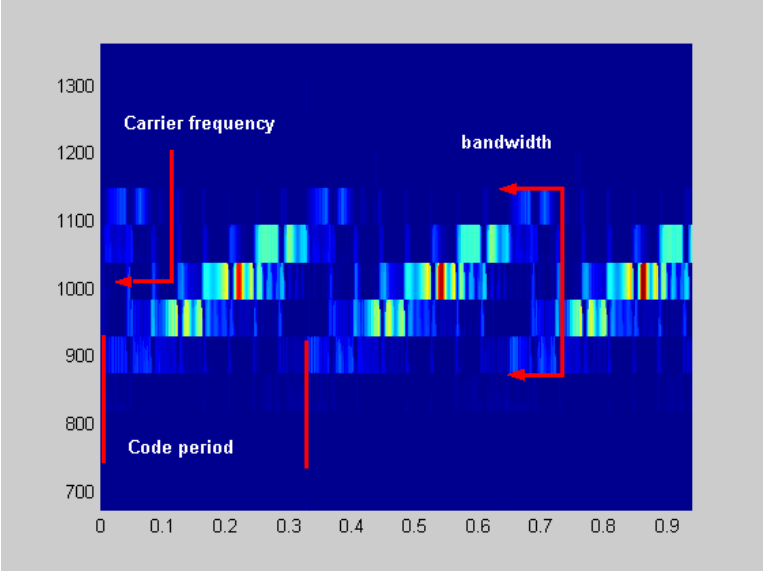
(a)



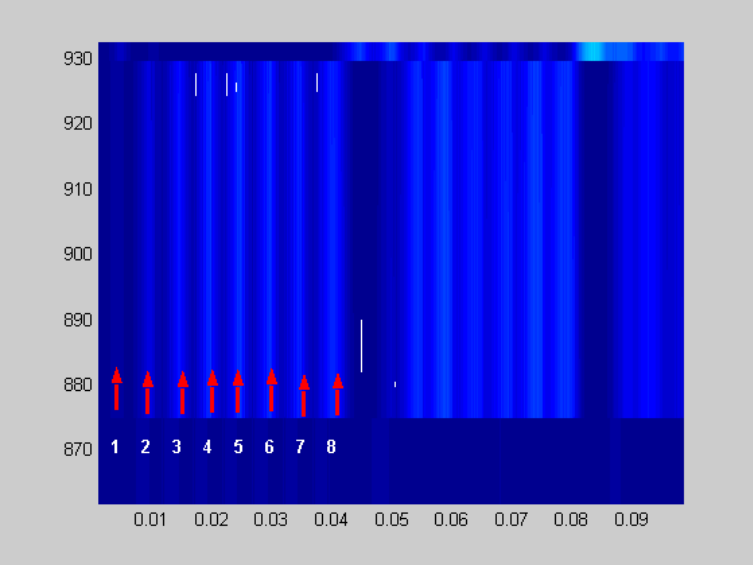
(b)

Figure 120 P1 Code *N*=64 cycles per phase =5 signal only (a) Output of the parallel filter arrays (b) Output after HOS.

Figure 121 (a) presents a zoom in the resulting plot after HOS. In this plot can be identified accurately carrier frequency, bandwidth and code period. Figure 121 (b) shows a zoom to recognize the phase shift in the signal. Eight phases can be observed; as a result, 64 phases are contained in one code period of the signal.



(a)



(b)

Figure 121 P1 Code *N*=64 cycles per phase =5 Signal only (a) Resulting plot after HOS showing parameters (b) First 8 phases from 16 in the signal.

2. P1 Code N=64 cycles per phase =5 SNR=0 dB

A P1 coded signal is described in Table 25 . With the objective to demonstrate the performance of the proposed signal processing to detect P1 coded signal in noise. Code period and bandwidth are calculated by the Equations (4.1.12) and (4.1.13). Figure 122 shows the PSD of the input signal.

P1 – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N - Phases	64	64	
SNR (dB)	0	-	
Cycles per phase	5	-	
Bandwidth	200	218.75	
Code Period (ms)	320	320	

Table 25 P1 Code *N*=64 cycles per phase =5 SNR=0 dB

The code period of the P1 signal is

$$t_c = \frac{(\text{Cycles/bit})(N)}{f_c} = \frac{(5)(64)}{1000} = 320 \text{ ms} \tag{4.1.12}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{5 \text{ cycles per phase}} = 200 \text{ Hz} \tag{4.1.13}$$

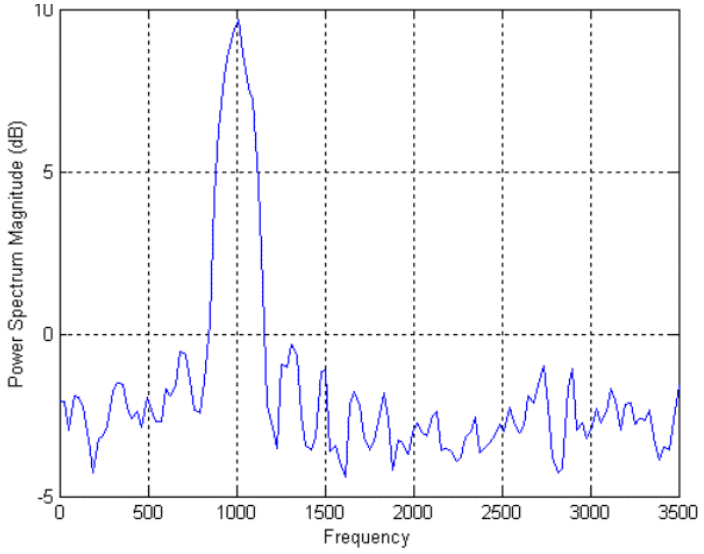
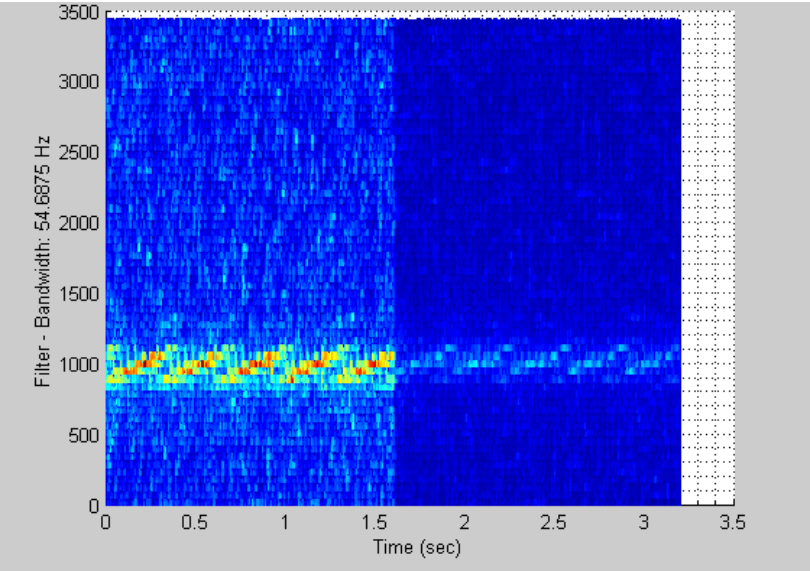
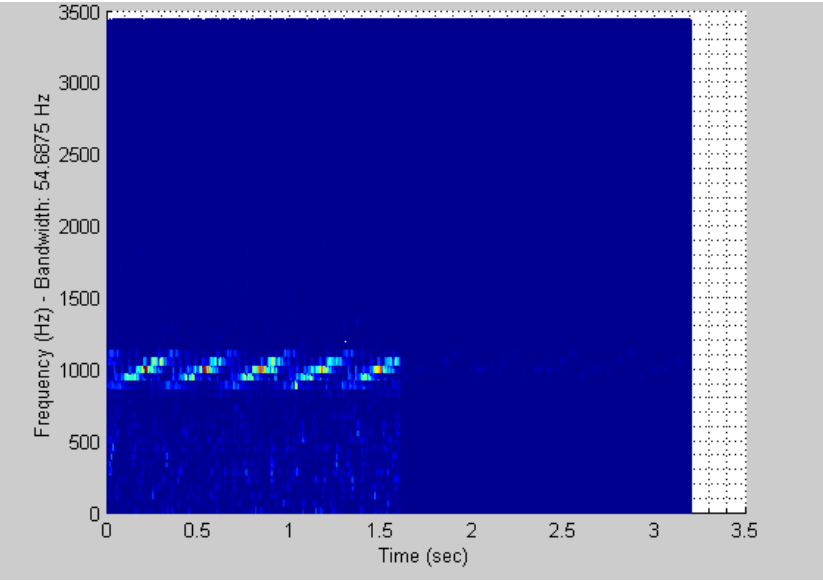


Figure 122 P1 Code *N*=64 cycles per phase =5 SNR=0 dB PSD.

The output signal of the parallel filter arrays is presented in Figure 123 (a). This is a frequency-time description of the signal where general characteristics of the signal can be observed. For an accurate estimation of the code period and bandwidth, it is necessary to obtain the third-order estimator of each sub-filter to eliminated the Gaussian noise in the signal as shown in Figure 123 (b).



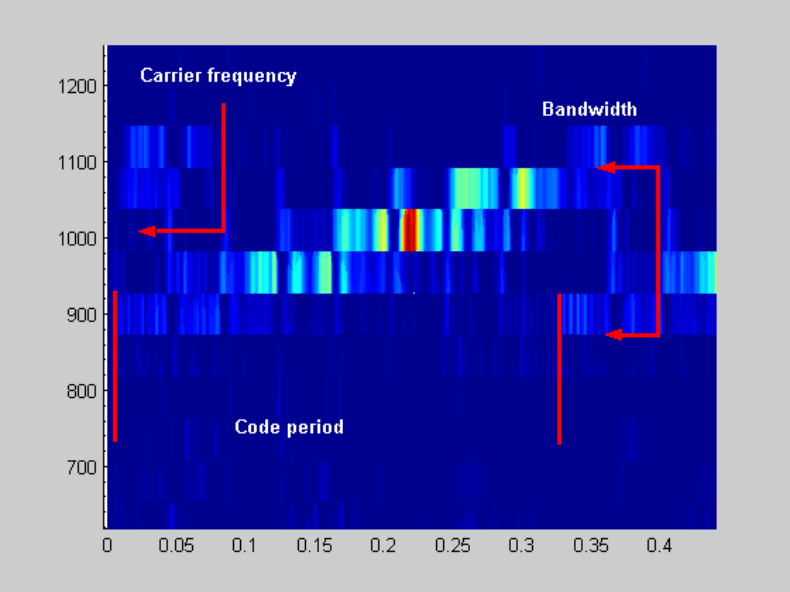
(a)



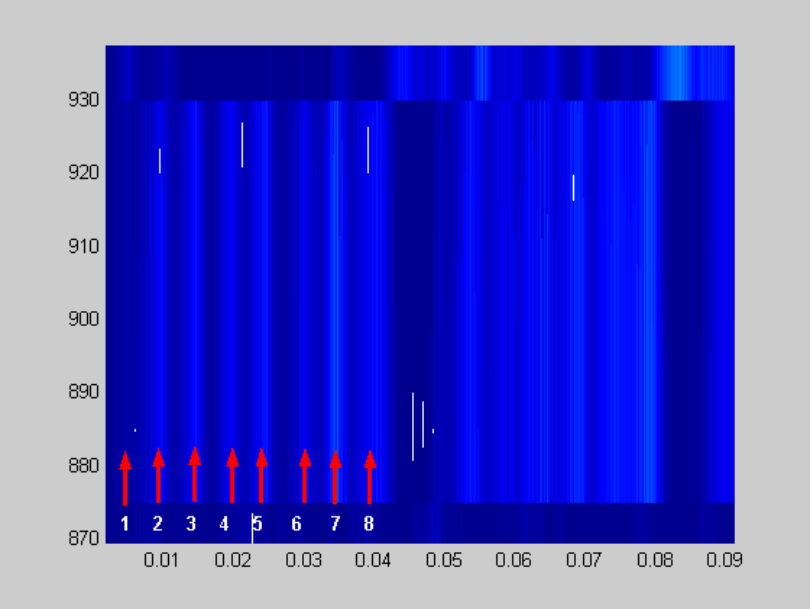
(b)

Figure 123 P1 Code N=64 cycles per phase =5 SNR=0 dB (a) Output of the parallel filter arrays (b) output after HOS.

The estimation of the carrier frequency, bandwidth and code period can be achieved making a zoom in the resulting plot after HOS as shown in Figure 124 (a). All this values are recorded and compared in the previous table. Figure 124 (b) illustrates the first 8 phases from a total of 64 present in the signal.

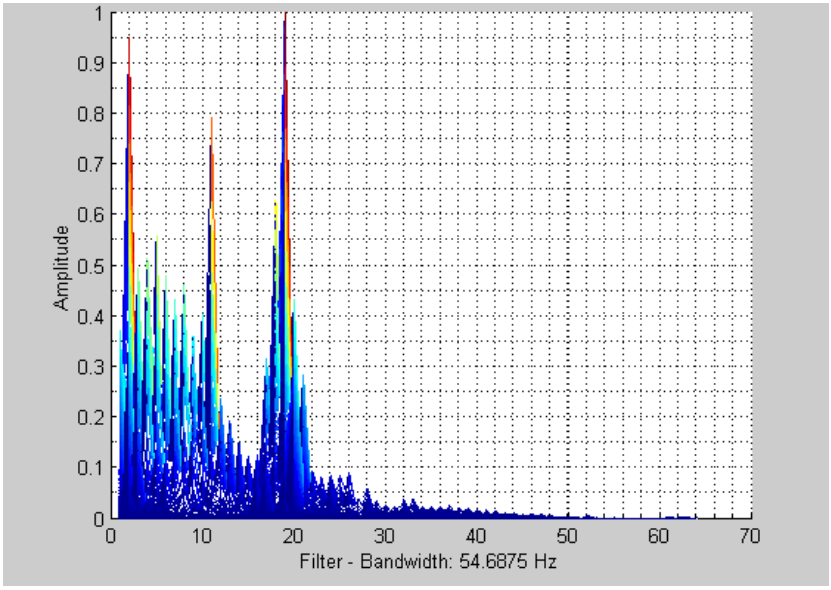


(a)

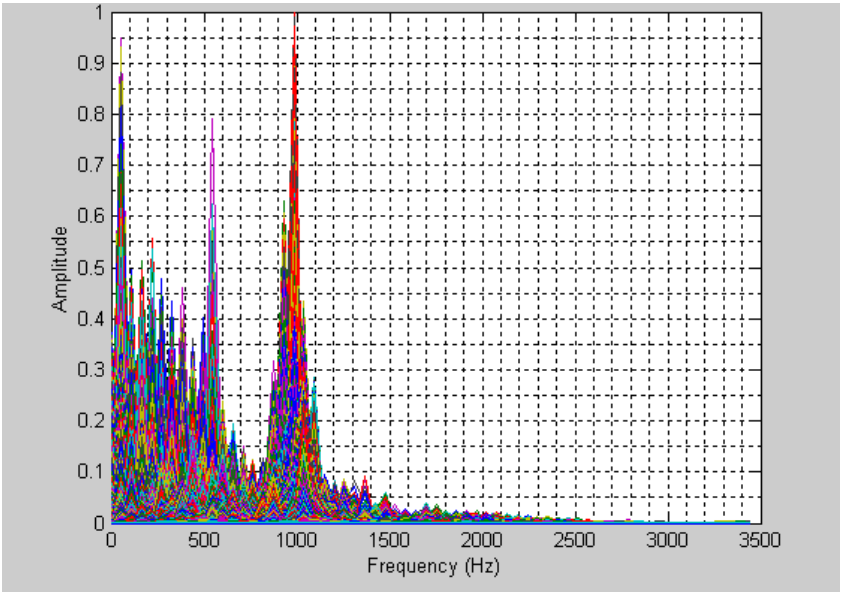


(b)

Figure 124 P1 Code *N*=64 cycles per phase =5 SNR=0 dB (a) Zoom in the resulting plot after HOS (b) First 8 of a total o 64 phases.



(a)



(b)

Figure 125 P1 Code *N*=64 cycles per phase =5 SNR=0 dB (a) Amplitude-filter plot (b) Amplitude-frequency plot.

3. Summary

Figure 126 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth, code period and number of phases in P1-coded signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very accurate values for carrier frequency, bandwidth and code period in any environment analyzed with a lower performance when SNR is less than –6 dB. The number of phases can be only extracted when noise is not presented and a number of cycles per phase is greater than 5.

P1	Carrier freq.(Hz)	Bandwidth (Hz)	Code period (ms)	Phases
Signal only	100%	107%	100%	100%
0 dB	100%	105%	100%	0%
(-) 6 dB	100%	121%	59%	0%

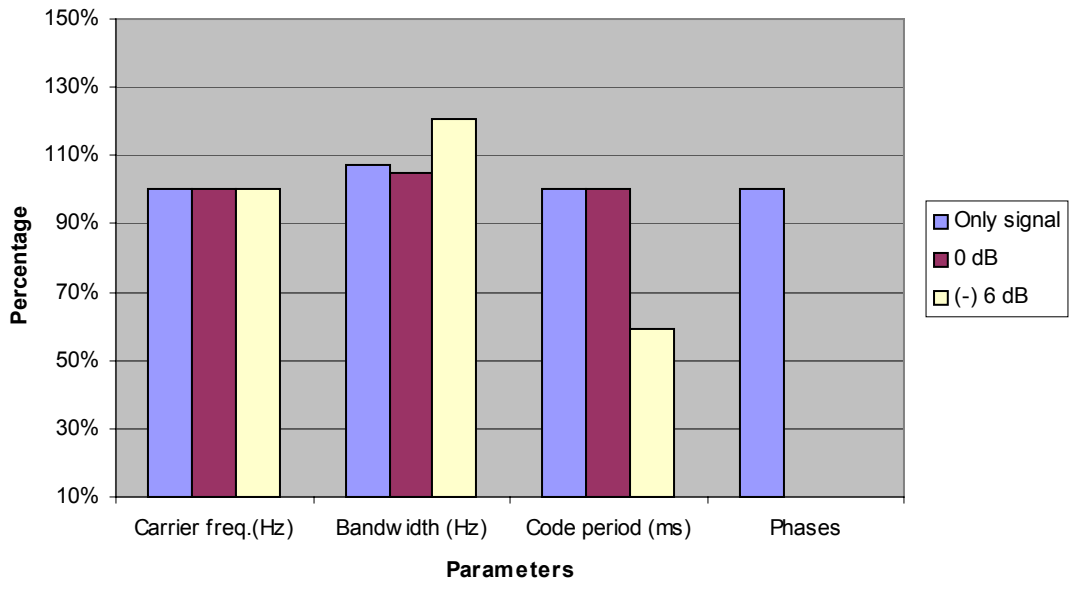


Figure 126 Performance of the signal processing detecting P1-coded signals.

F. P2 POLYPHASE CODE

This code is essentially derived in the same way as the P1 code. The P2 code has the same phase increments within each group as the P1 code, except that the starting phase is different. This code has the frequency symmetry of the P1 code while also containing the property of having phases symmetric in the center of the code. The P2 polyphase code, as well as the P1, has more of a symmetrical spectrum than a Frank-coded signal due to its symmetry in the carrier.

A set of 12 different signals was generated to demonstrate the effectiveness of the parallel filter arrays and HOS on the detection of this polyphase-coded signal. This signals are show in Table 26 . Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	File	Carrier Frequency	Sampling Frequency	N(Phases)	Cycles/phase	SNR
1	P2_1_7_4_1_s	1000	7000	4	1	-
2	P2_1_7_4_1_0	1000	7000	4	1	0
3	P2_1_7_4_1_-6	1000	7000	4	1	-6
4	P2_1_7_4_5_s	1000	7000	4	5	-
5	P2_1_7_4_5_0	1000	7000	4	5	0
6	P2_1_7_4_5_-6	1000	7000	4	5	-6
7	P2_1_7_8_1_s	1000	7000	8	1	-
8	P2_1_7_8_1_0	1000	7000	8	1	0
9	P2_1_7_8_1_-6	1000	7000	8	1	-6
10	P2_1_7_8_5_s	1000	7000	8	5	-
11	P2_1_7_8_5_0	1000	7000	8	5	0
12	P2_1_7_8_5_-6	1000	7000	8	5	-6

Table 26 Matrix of test signals for P2 polyphase code.

1. P2 Code, N=16, cycles per phase =5 and signal only

Table 27 shows a P2-coded signal with carrier frequency equal to 1000 Hz, sampling frequency equal to 7000 Hz, 16 phases and 5 cycles per phase. Code period and bandwidth are calculated by the Equations (4.1.14) and (4.1.15). Figure 127 (a) shows the PSD of the input signal illustrating the distribution of the power along the different frequencies contained in the signal. Figure 127 (b) presents a portion of the time-domain plot.

P2 – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N – Phases	16	16	
SNR (dB)	Signal only	-	
Cycles per phase	5	-	
Bandwidth	200	218.75	
Code Period (ms)	80	80	

Table 27 P2 Code N=16 cycles per phase =5 signal only.

The code period of the P2 signal is

$$t_c = \frac{(\text{Cycles/bit})(N)}{f_c} = \frac{(5)(16)}{1000} = 80 \text{ ms} \tag{4.1.14}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{5 \text{ cycles per phase}} = 200 \text{ Hz} \tag{4.1.15}$$

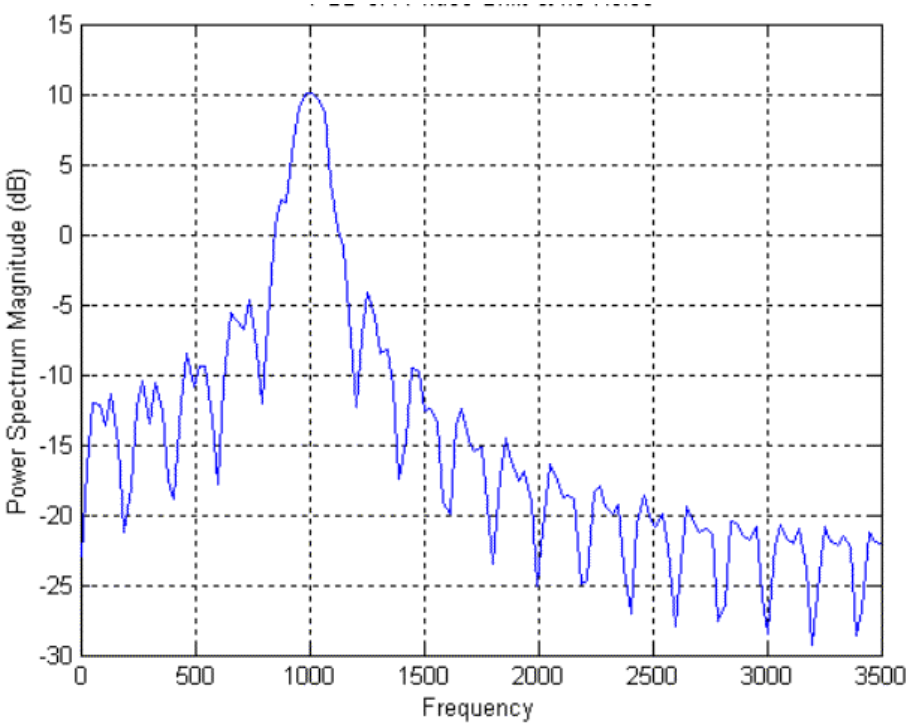
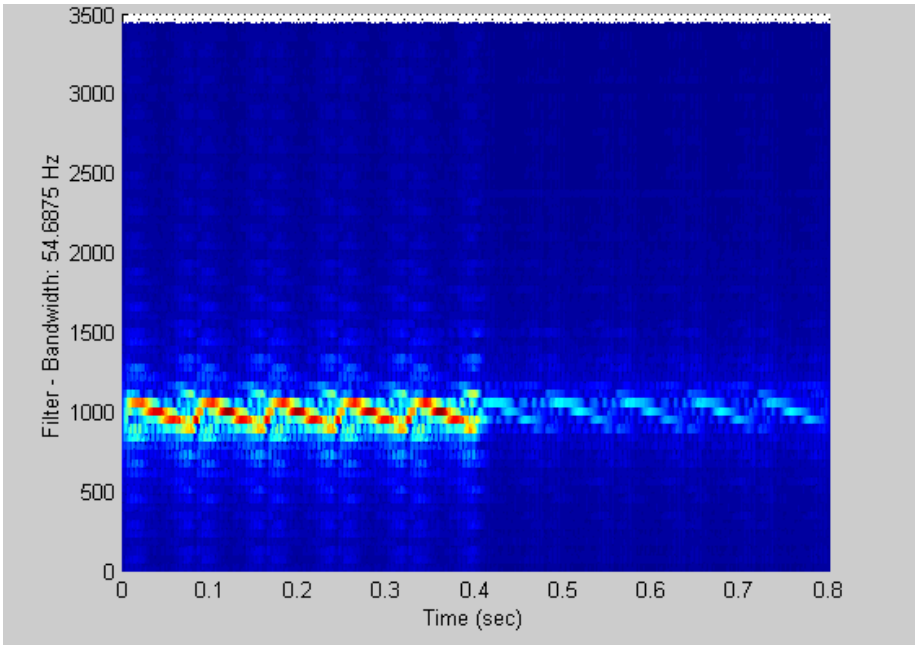


Figure 127 P2 Code *N*=16 cycles per phase =5 Signal only PSD.

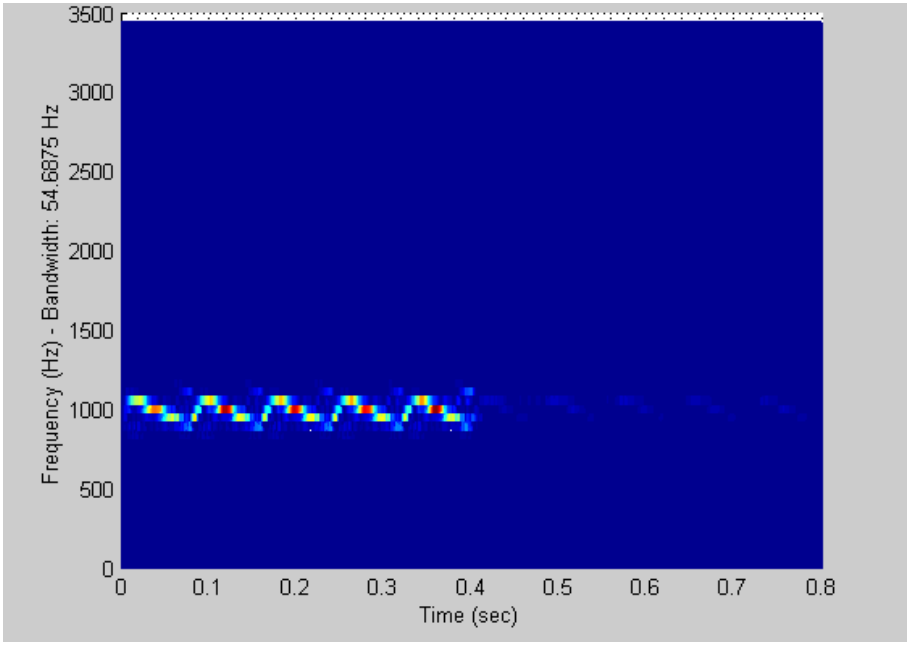
The response of the parallel filter arrays is illustrated in Figure 128 (a). This figure provides a time-frequency description and representation of the signal. The signal is decomposes the input signal into sub-band signals with narrow frequency bands. As said in Chapter 3, the filter bank can be interpreted as a matrix *LxT*, where *L* represents the number of filters and T time. In this case, 64 filters forms the parallel filter arrays and 0.8 seconds of the signal are analyzed.

Figure 128 (b) shows the output signal after third-order cumulant estimators are applied to each sub-band signal. Figure 129 (a) and (b) shows a zoom in the previous plot. Carrier frequency, bandwidth, code period and numer of phases are measured, and the results are compared with the input parameters in the previous table.

Figure 130 provides an amplitude-frequency representation of the resulting signal. The plot is very useful to confirm parameters, such as carrier frequency and bandwidth.

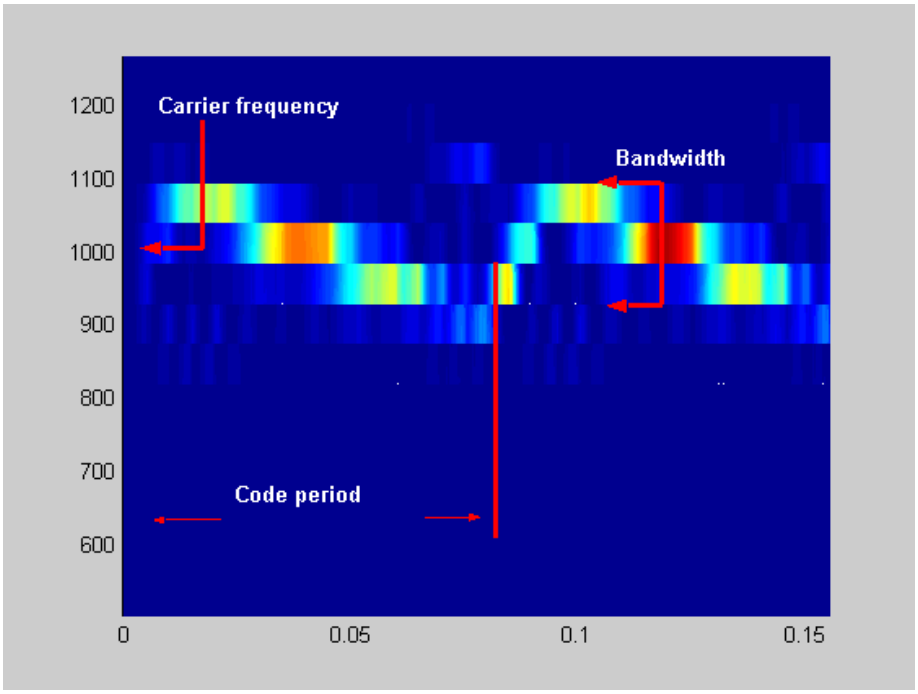


(a)

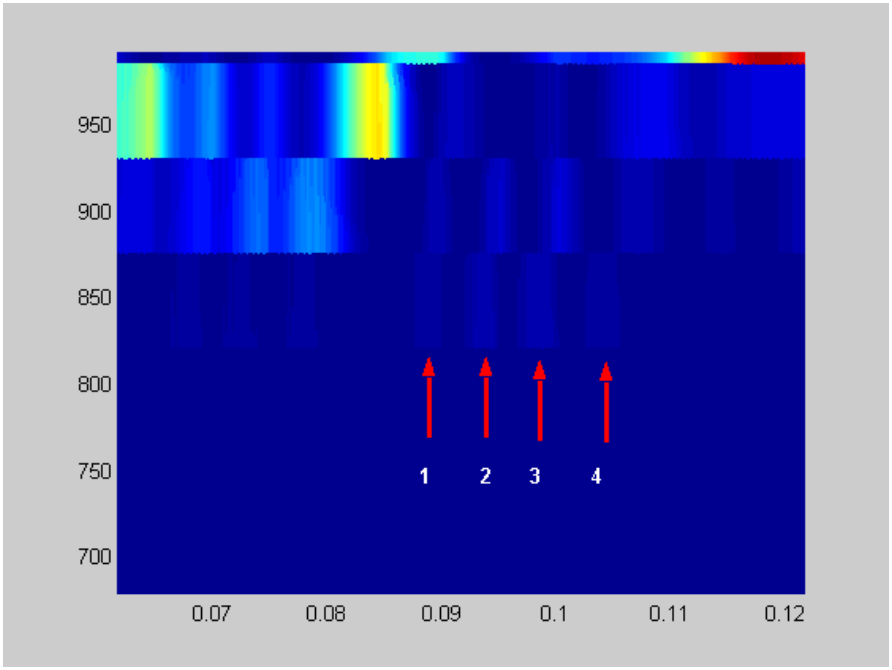


(b)

Figure 128 P2 Code N=16 cycles per phase =5 signal only (a) output of the parallel filter arrays (b) output after HOS.

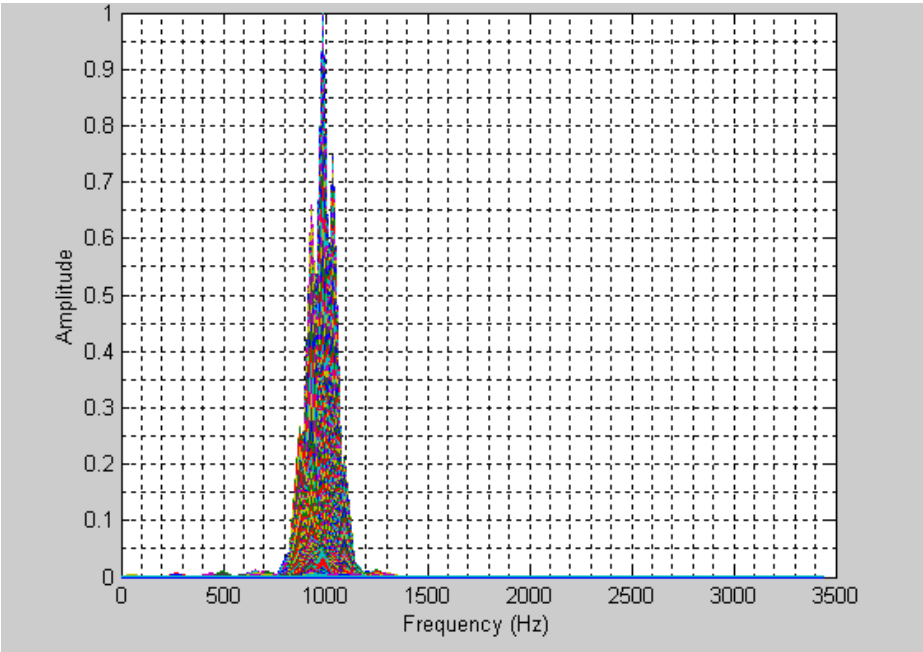


(a)



(b)

Figure 129 P2 Code *N*=16 cycles per phase =5 signal only (a) zoom in previous plot (b)) Plot showing 4 phases of 16.



(b)

Figure 130 P2 Code *N*=16 cycles per phase =5 signal only, amplitude-frequency plot.

2. P2 Code, N=16, cycles per phase =5 and SNR= 0 dB

Table 28 shows a P2-coded signal with carrier frequency equal to 1000 Hz, sampling frequency equal to 7000 Hz, 16 phases and 5 cycles per phase. Code period and bandwidth are calculated by the Equations (4.1.16) and (4.1.17). Figure 131 shows the PSD of the input signal illustrating the distribution of the power along the different frequencies contained in the signal.

P2 – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N – Phases	16	16	
SNR (dB)	0	-	
Cycles per phase	5	-	
Bandwidth	200	218.75	
Code Period (ms)	80	80	

Table 28 P2 Code *N*=16 cycles per phase =5 SNR= 0 dB.

The code period of the P2 signal is

$$t_c = \frac{(\text{Cycles/bit})(N)}{f_c} = \frac{(5)(16)}{1000} = 80 \text{ ms} \tag{4.1.16}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{5 \text{ cycles per phase}} = 200 \text{ Hz} \tag{4.1.17}$$

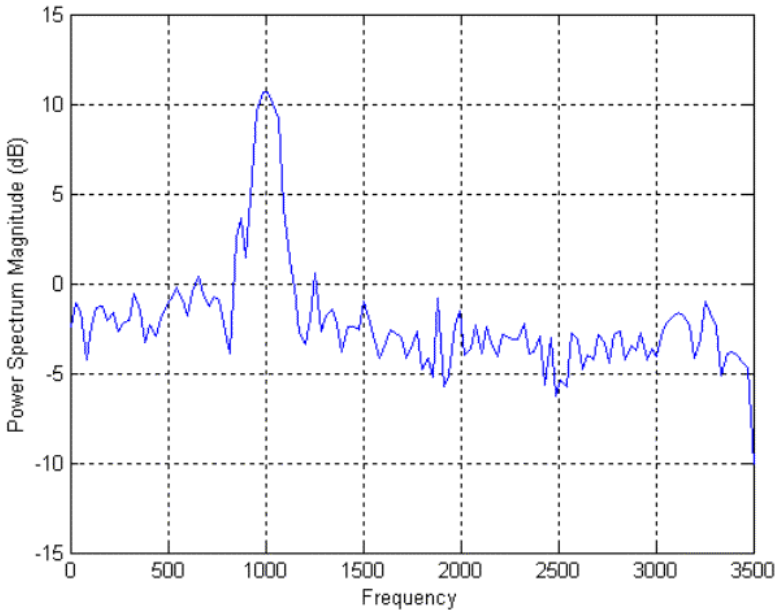


Figure 131 P2 Code *N*=16 cycles per phase =5 SNR= 0 dB PSD.

Figure 133 shows the response of the parallel filter arrays. This figure provides a time-frequency description and representation of the signal. The signal is decomposes the input signal into sub-band signals with narrow frequency bands.

Figure 135 illustrates the output signal after third-order cumulant estimators are applied to each sub-band signal to suppress the noise. This plot presents the performance of the HOS to eliminate white Gaussian noise and extract enough information for the recognition of the signal. Figure 134 shows a zoom in the previous plot. Carrier frequency, bandwidth, and code period are measured and the results are compared with the input parameters in the previous table. Additionally, Figure 135 shows four phases in one of the frequencies. Four groups of four phases can be detected totalizing 16 phases in each period of the signal.

Figure 136 provides an amplitude-frequency representation of the resulting signal. These plots are very useful to confirm parameters, such as carrier frequency and bandwidth.

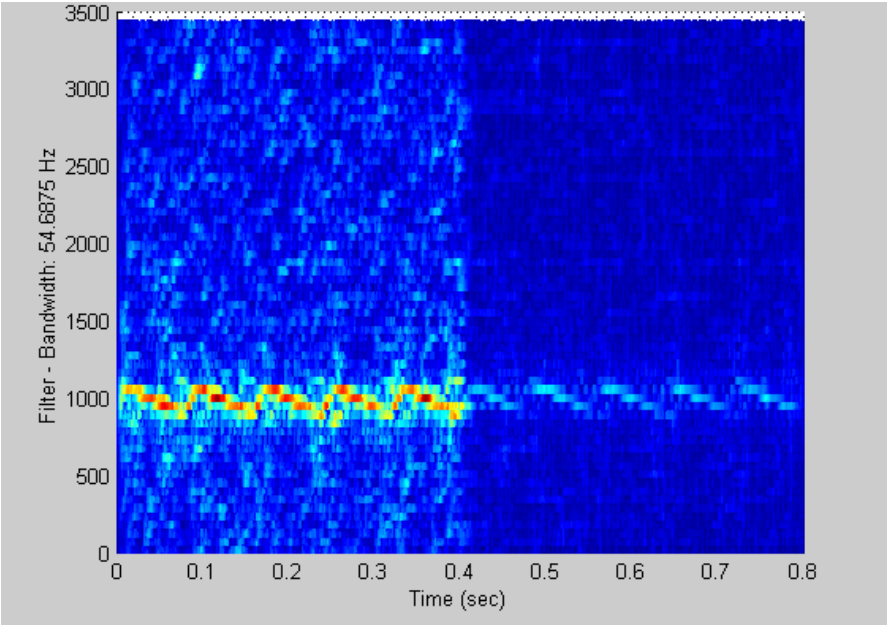


Figure 132 P2 Code *N*=16 cycles per phase =5 SNR= 0 dB output of the parallel filter arrays

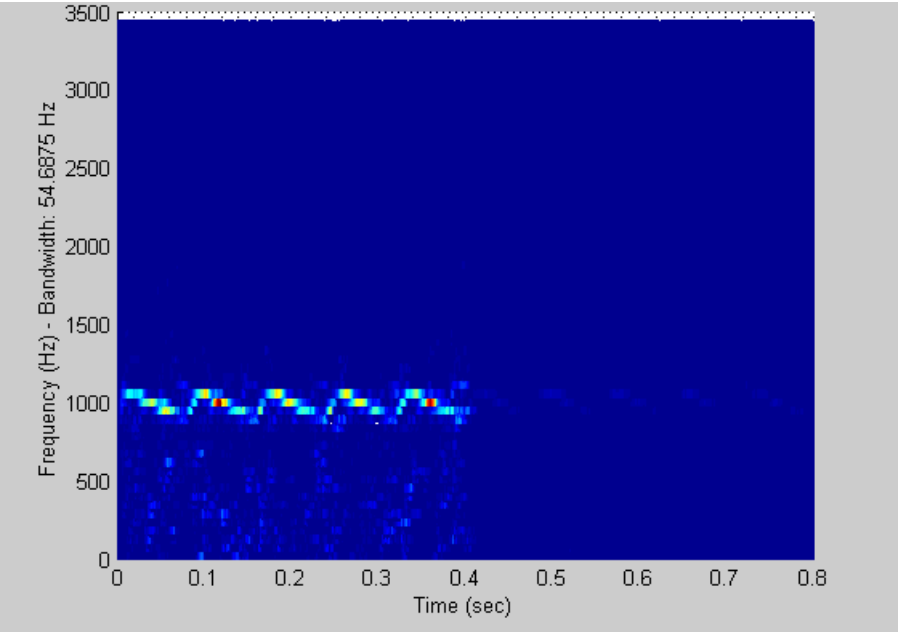


Figure 133 P2 Code *N*=16 cycles per phase =5 SNR= 0 dB output after HOS.

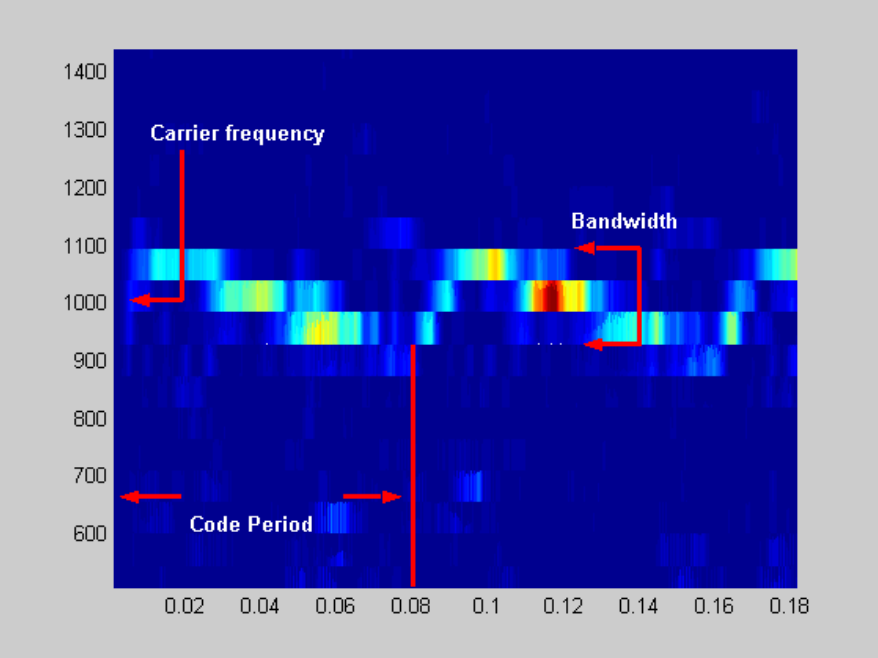


Figure 134 P2 Code *N*=16 cycles per phase =5 SNR= 0 dB zoom in the resulting plot after HOS.

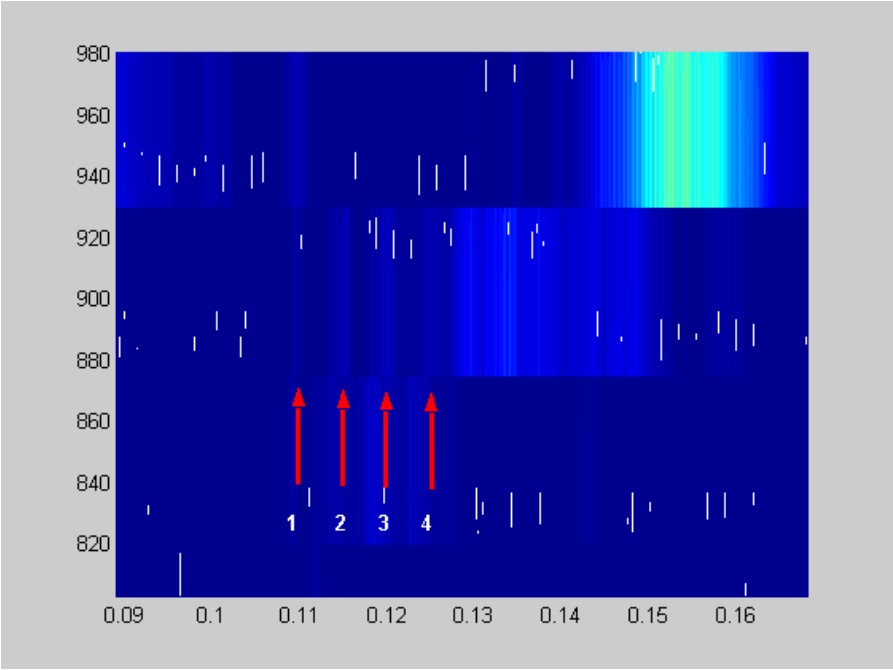
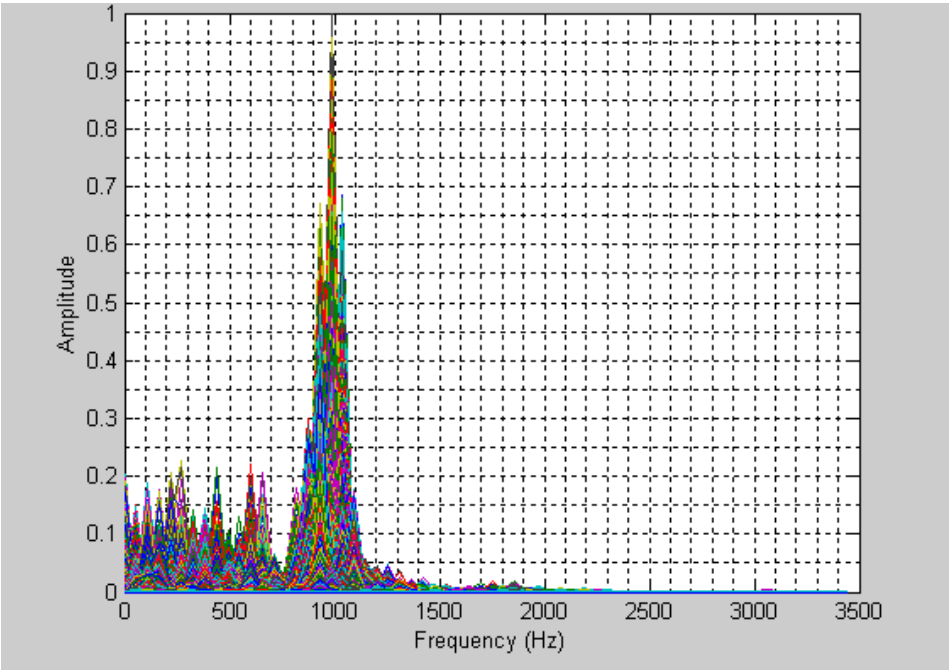


Figure 135 P2 Code *N*=16 cycles per phase =5 SNR= 0 dB plot showing 4 phases of 16.



(b)

Figure 136 P2 Code *N*=16 cycles per phase =5 SNR= 0, amplitude-frequency plot.

3. Summary

Figure 137 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth, code period and number of phases in P2-coded signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very accurate values for carrier frequency, bandwidth and code period in any environment analyzed with a lower performance when SNR is less than –6 dB. The number of phases can be only extracted when noise is not presented and a number of cycles per phase is greater than 5.

P2	Carrier freq.(Hz)	Bandwidth (Hz)	Code period (ms)	Phases
Signal only	100%	100%	100%	100%
0 dB	100%	100%	100%	0%
(-) 6 dB	100%	121%	97%	0%

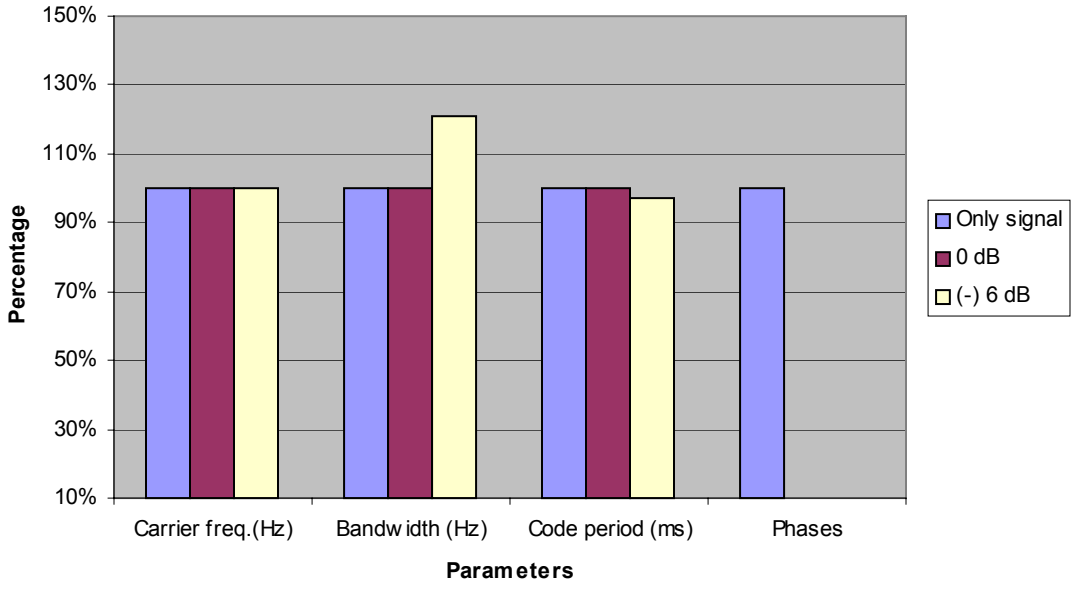


Figure 137 Performance of the signal processing detecting P2-coded signals.

G. P3 POLYPHASE CODE

Chapter 2 describes in the characteristics of P3 polyphase codes. The P3 code only differs from Frank code by 180 degrees phase shifts every $N^{1/2}$ code elements (one frequency group) and by added phase increments that repeat every $N^{1/2}$ samples (every frequency group). The P3 polyphase codes, as well as P4 codes, have tolerance to Doppler by the lack of grating side lobes. The performance of a parallel filter arrays and HOS to detect P3-coded signals is evaluated in this section.

Twelve different P3-coded test signals were generated varying the number of phases, the number of cycles per phase and SNR as shown in Table 29 . Input parameters, calculation and results are presented and compared in tables. Plots describe the input signals and provide a graphical representation of the results. Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	File	Carrier Frequency	Sampling Frequency	N(Phases)	Cycles/phase	SNR
1	P3_1_7_16_1_s	1000	7000	4	1	-
2	P3_1_7_16_1_0	1000	7000	4	1	0
3	P3_1_7_16_1_-6	1000	7000	4	1	-6
4	P3_1_7_16_5_s	1000	7000	4	5	-
5	P3_1_7_16_5_0	1000	7000	4	5	0
6	P3_1_7_16_5_-6	1000	7000	4	5	-6
7	P3_1_7_64_1_s	1000	7000	8	1	-
8	P3_1_7_64_1_0	1000	7000	8	1	0
9	P3_1_7_64_1_-6	1000	7000	8	1	-6
10	P3_1_7_64_5_s	1000	7000	8	5	-
11	P3_1_7_64_5_0	1000	7000	8	5	0
12	P3_1_7_64_5_-6	1000	7000	8	5	-6

Table 29 Matrix of test signals for P3 Polyphase code.

1. P3 Code, N=64, cycles per phase =1 and signal only

Table 30 shows a P3-coded signal with carrier frequency equal to 1000 Hz, sampling frequency equal to 7000 Hz, 64 phases and 1 cycles per phase. Code period and bandwidth are calculated by the Equations (4.1.18) and (4.1.19). Figure 138 (a) shows the PSD of the input signal, illustrating the distribution of the power along the different frequencies contained in the signal. Figure 138 (a) illustrates the phase shift for a P3-coded signal with N=64 and showing the symmetry at the center frequency.

P3 – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N – Phases	64	64	
SNR (dB)	Signal only	-	
Cycles per phase	1	1	
Bandwidth	1000	1039	
Code Period (ms)	64	64	

Table 30 P3 Code *N*=64 cycles per phase =1 signal only.

The code period of the P3 signal is

$$t_c = \frac{(\text{Cycles/bit})(N)}{f_c} = \frac{(1)(64)}{1000} = 64 \text{ ms} \tag{4.1.18}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{1 \text{ cycles per phase}} = 1000 \text{ Hz} \tag{4.1.19}$$

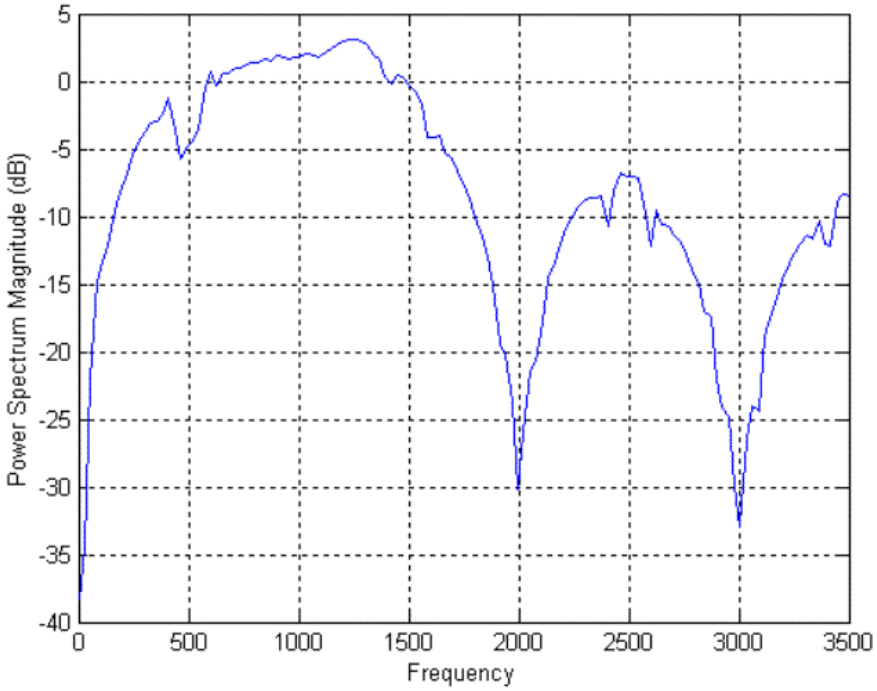
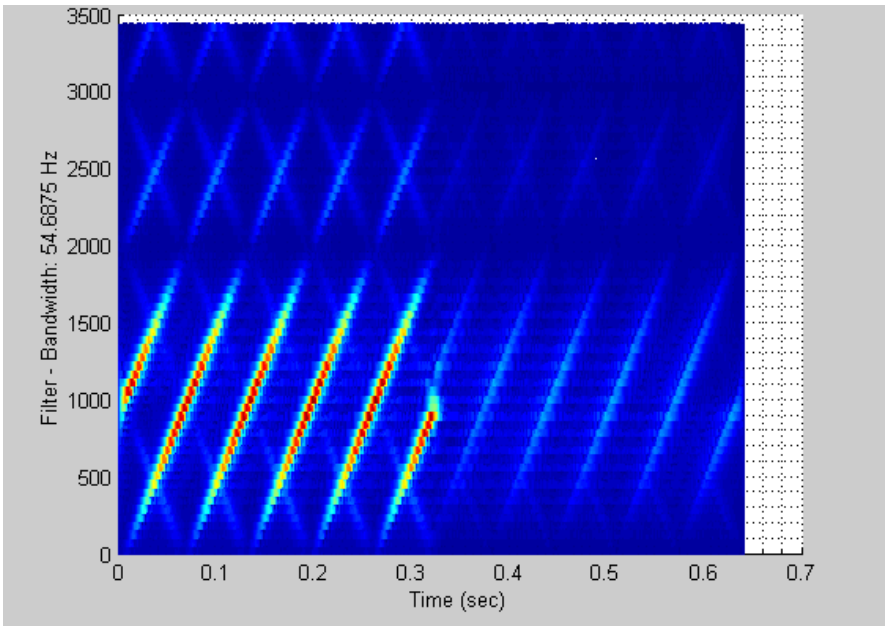


Figure 138 P3 Code *N*=64 cycles per phase =1 signal only PSD.

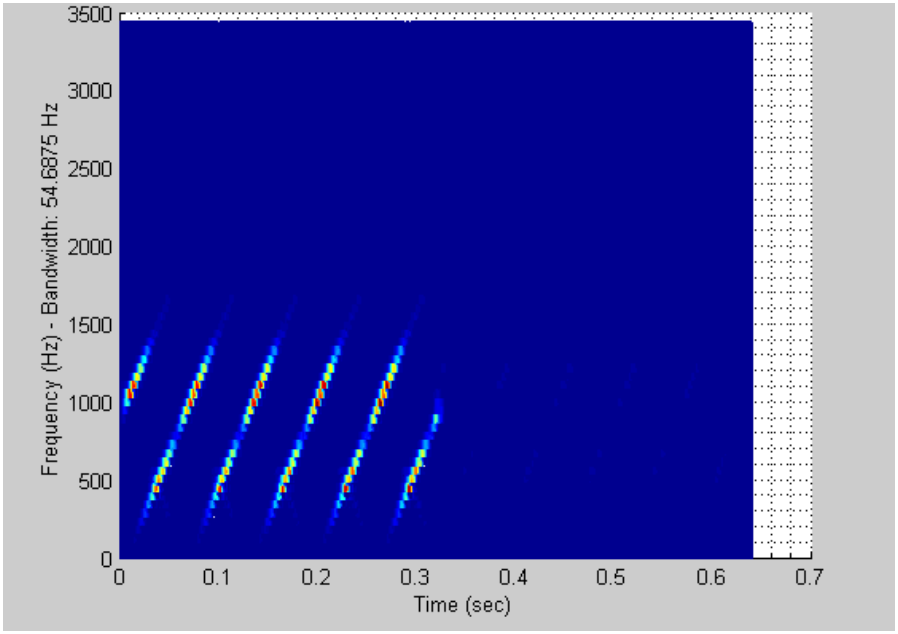
The response of the parallel filter arrays is illustrated in Figure 139 (a). Similarly as in previous analysis, this filter bank separates the input signal in smaller sub-bands, providing a time-frequency description of the signal. Additionally, this figure shows the symmetry of the phase shifts at the center frequency.

A third-order cumulant estimator is applied to each one of the filters and the output signal is shown in Figure 139 (b). The resulting signal doesn't suffer a dramatic change due to the nonexistence of noise added to this test signal. A zoom in this plot is presented in Figure 140 (a), where carrier frequency, bandwidth and code period are measured and compared with the input values in the previous table.

In addition, Figure 140 (b) presents an amplitude-frequency plot confirming some of the previous measurements.

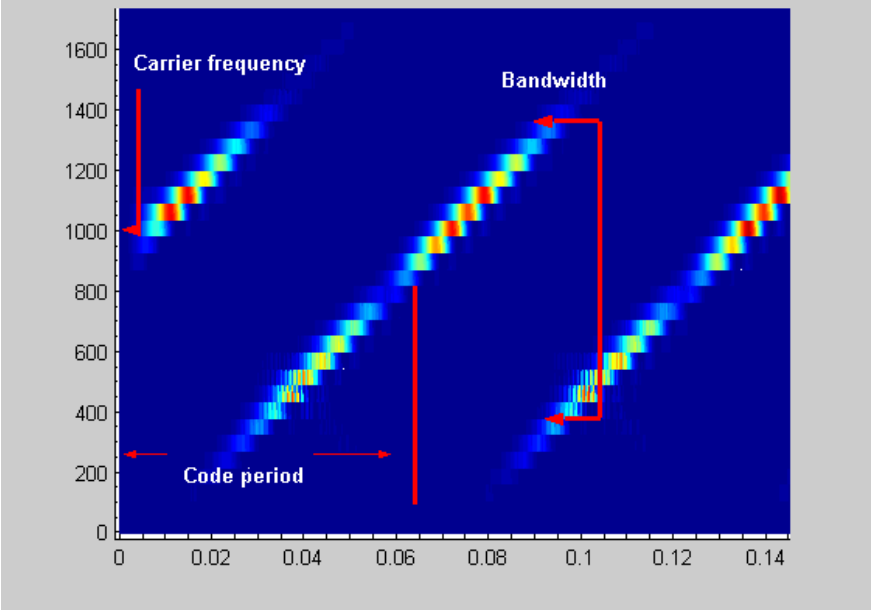


(a)

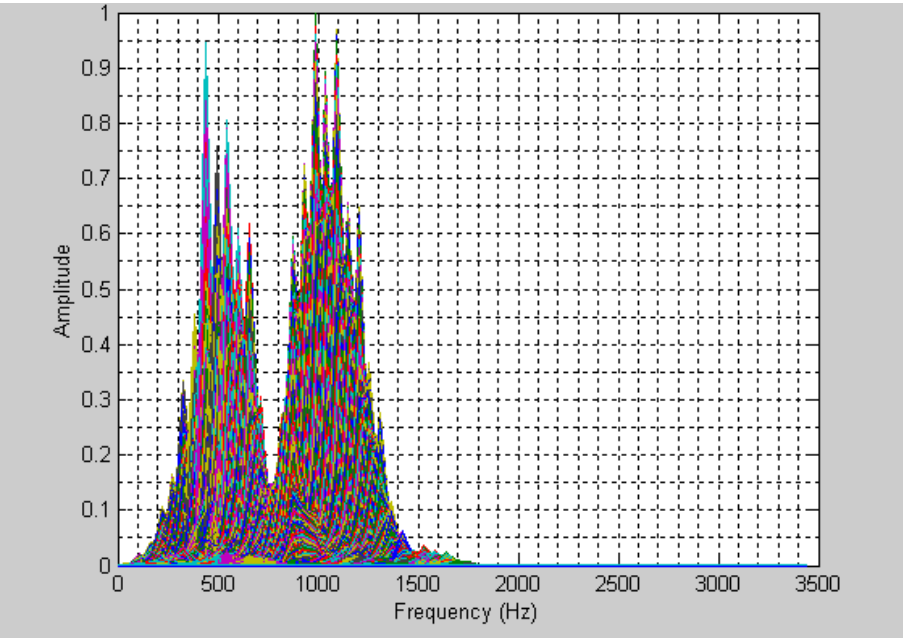


(b)

Figure 139 P3 Code *N*=64 cycles per phase =1 signal only (a) Output of the parallel filter arrays (b) Output after HOS.



(a)



(b)

Figure 140 P3 Code *N*=64 cycles per phase =1 signal only (a) zoom in the resulting signal after HOS (b) Amplitude-frequency plot.

2. P3 Code, N=64, cycles per phase =1 and SNR=0 dB

Table 31 shows a P3-coded signal with carrier frequency equal to 1000 Hz, sampling frequency equal to 7000 Hz, 64 phases, 1 cycles per phase and SNR = 0 dB. Code period and bandwidth are calculated by the Equations (4.1.20) and (4.1.21). Figure 141 shows the PSD of the input signal, illustrating the distribution of the power along the different frequencies contained in the signal.

P3 – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N – Phases	64	-	
SNR (dB)	0	-	
Cycles per phase	1	1	
Bandwidth	1000	1039	
Code Period (ms)	64	64	

Table 31 P3 Code *N*=64 cycles per phase =1 SNR=0 dB.

The code period of the P3 signal is

$$t_c = \frac{(\text{Cycles/bit})(N)}{f_c} = \frac{(1)(64)}{1000} = 64 \text{ ms} \tag{4.1.20}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{1 \text{ cycles per phase}} = 1000 \text{ Hz} \tag{4.1.21}$$

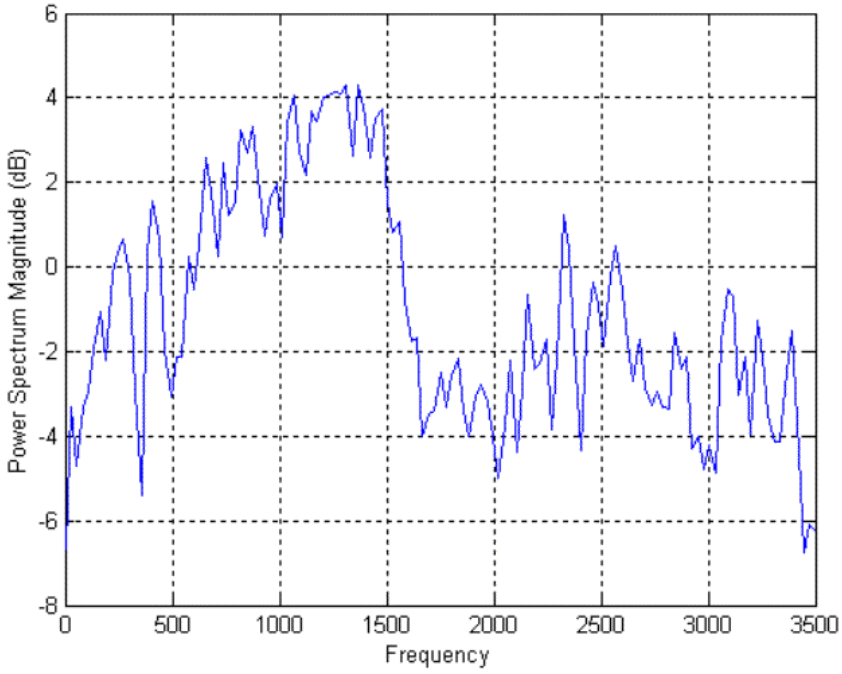
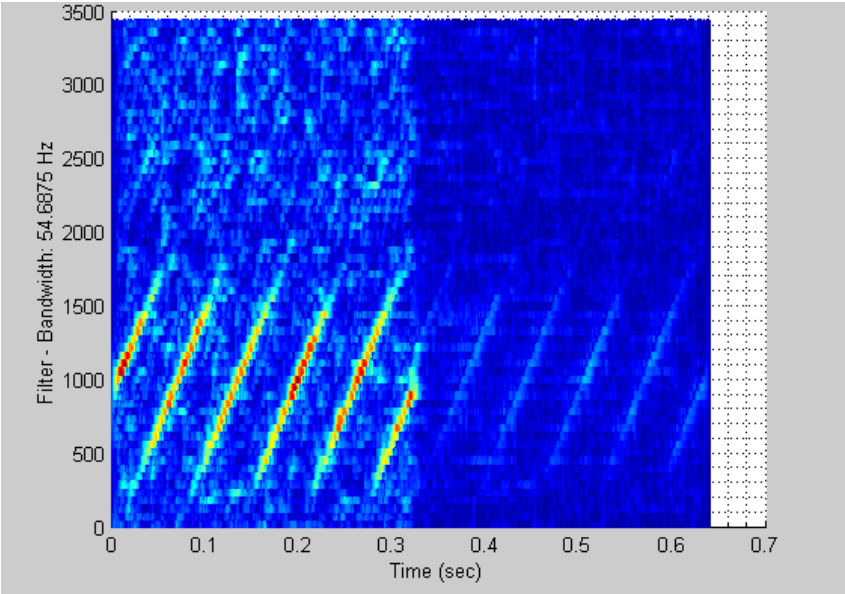


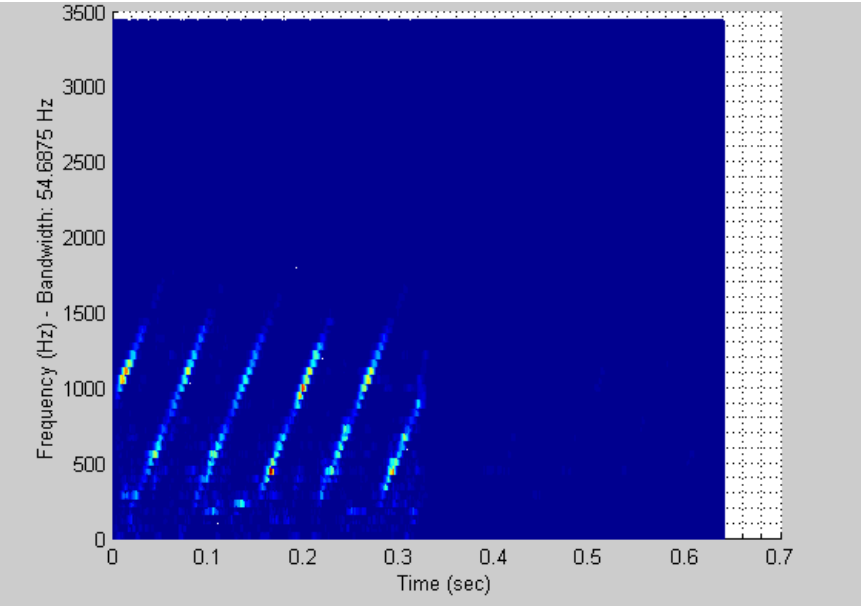
Figure 141 P3 Code *N*=64 cycles per phase =1 SNR=0 dB PSD.

The performance of the method based in filter bank and HOS is tested in this example where the SNR = 0 dB. Figure 142 (a) shows the response of the parallel filter arrays for the P3-coded signal described above. The signal is decomposed into subband signals with narrow frequency bands. The structure of the parallel filter arrays can be interpreted as a matrix LxT where L is the number of sub-filter and T is time. Even though SNR= 0 dB, this filter bank provides a clear frequency-time description of the input signal. Once the signal is separated into small sub-band signals, a third-order cumulant estimator is applied to each one of the sub-band signals. Figure 142 (b) illustrates the resulting signal after HOS, where the white Gaussian noise was suppressed. Because of some inter-modulation products, low-frequency signals appear in the plots. With SNR = 0 dB, the introduced noise doesn't interfere with the detection problem.

Figure 143 (a) shows a zoom in the resulting signal after HOS. Carrier frequency, bandwidth and code period are measured and compared in the previous table. Figure 143 (b) provides an amplitude-frequency view of the resulting signal.

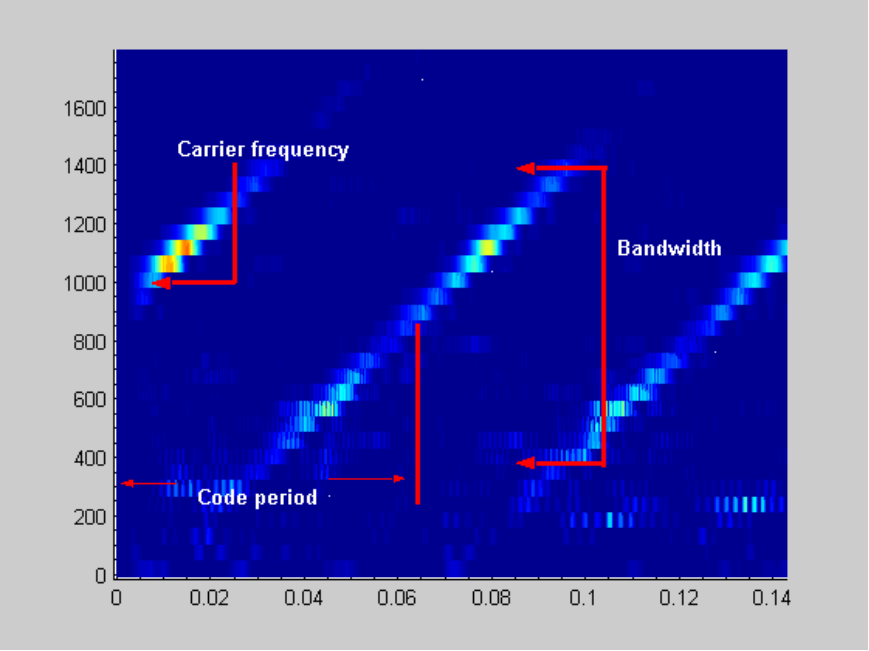


(a)

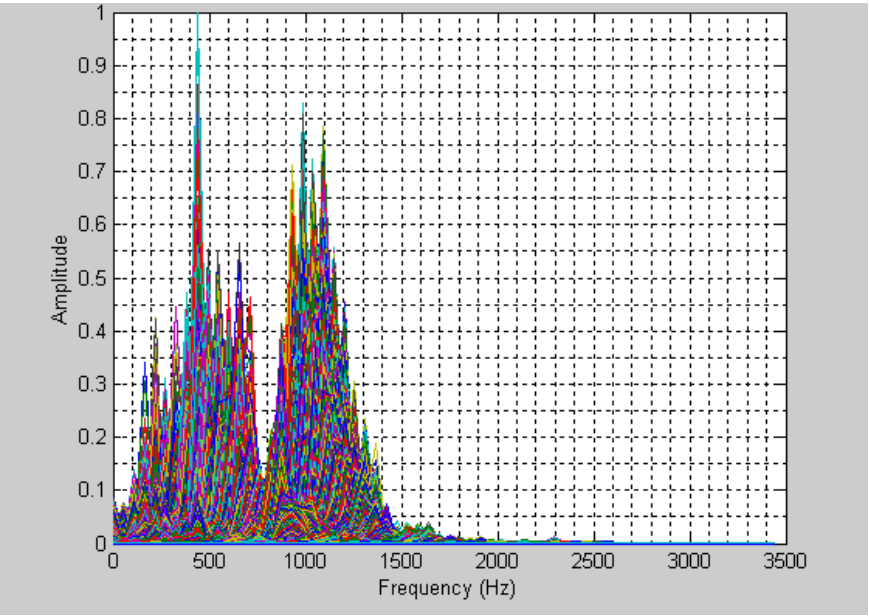


(b)

Figure 142 P3 Code *N*=64 cycles per phase =1 SNR=0 dB (a) Output of the parallel filter arrays (b) Output after HOS.



(a)



(b)

Figure 143 P3 Code N=64 cycles per phase =1 SNR=0 dB (a) Zoom in the resulting plot after HOS (b) Amplitude-frequency plot.

3. Summary

Figure 144 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth, code period and number of phases in P3-coded signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very accurate values for carrier frequency, bandwidth and code period in any environment analyzed with a lower performance when SNR is less than –6 dB. The number of phases can be only extracted when noise is not presented and a number of cycles per phase is greater than 5.

P3	Carrier freq.(Hz)	Bandwidth (Hz)	Code period (ms)	Phases
Signal only	100%	100%	100%	100%
0 dB	100%	100%	100%	0%
(-) 6 dB	100%	103%	97%	0%

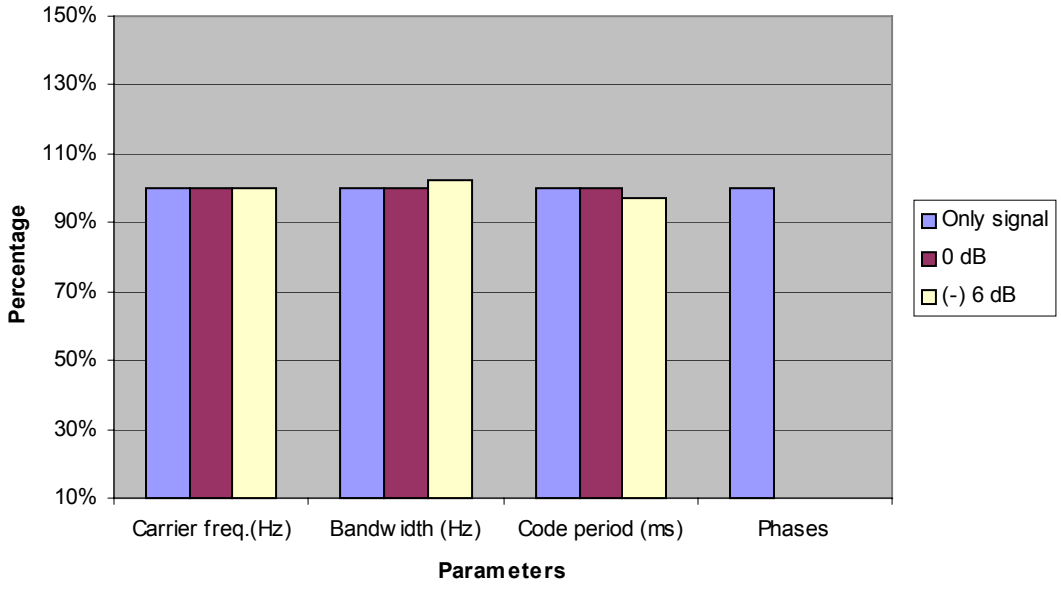


Figure 144 Performance of the signal processing detecting P3-coded signals.

H. P4 POLYPHASE CODE

The P4 code is conceptually derived from the same waveform as P3 code. P4 code consists of the discrete phases of the linear chirp waveform taken at specific time intervals and exhibits the same range Doppler coupling associated with the chirp waveform. However, the peak sidelobe levels are lower than those of the unweighted chirp waveform.

The effectiveness of the proposed method on the detection and identification of unknown P4-coded signals is proven in this section. Twelve different P4-coded test signals were generated varying the number of phases, the number of cycles per phase and SNR as shown in Table 32 .

Input parameters, calculation and results are presented and compared in tables. Plots describe the input signals and provide a graphical representation of the results. Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	File	Carrier Frequency	Sampling Frequency	N(Phases)	Cycles/phase	SNR
1	P4_1_7_16_1_s	1000	7000	4	1	-
2	P4_1_7_16_1_0	1000	7000	4	1	0
3	P4_1_7_16_1_-6	1000	7000	4	1	-6
4	P4_1_7_16_5_s	1000	7000	4	5	-
5	P4_1_7_16_5_0	1000	7000	4	5	0
6	P4_1_7_16_5_-6	1000	7000	4	5	-6
7	P4_1_7_64_1_s	1000	7000	8	1	-
8	P4_1_7_64_1_0	1000	7000	8	1	0
9	P4_1_7_64_1_-6	1000	7000	8	1	-6
10	P4_1_7_64_5_s	1000	7000	8	5	-
11	P4_1_7_64_5_0	1000	7000	8	5	0
12	P4_1_7_64_5_-6	1000	7000	8	5	-6

Table 32 Matrix of test signals for P4 Polyphase code.

1. P4 Code, N=64, cycles per phase =5 and signal only

Table 33 shows a P4-coded signal with carrier frequency equal to 1000 Hz, sampling frequency equal to 7000 Hz, 64 phases and 5 cycles per phase. Code period and bandwidth are calculated by the Equations (4.1.22) and (4.1.23). Figure 145 (a) shows the PSD of the input signal, illustrating the distribution of the power along the different frequencies contained in the signal. Figure 145 (b) illustrates the phase shifts for a P4-coded signal with N=64.

P4 – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N – Phases	64	64	
SNR (dB)	Signal only	-	
Cycles per phase	5	5	
Bandwidth	200	218.75	
Code Period (ms)	320	320	

Table 33 P4 Code N=64 cycles per phase =5 signal only.

The code period of the P4 signal is

$$t_c = \frac{(\text{Cycles/bit})(N)}{f_c} = \frac{(5)(64)}{1000} = 320 \text{ ms} \tag{4.1.22}$$

The bandwidth of the signal depends on the cycles per phase (or chirp) as

$$B = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{2 \text{ cycles per phase}} = 200 \text{ Hz} \tag{4.1.23}$$

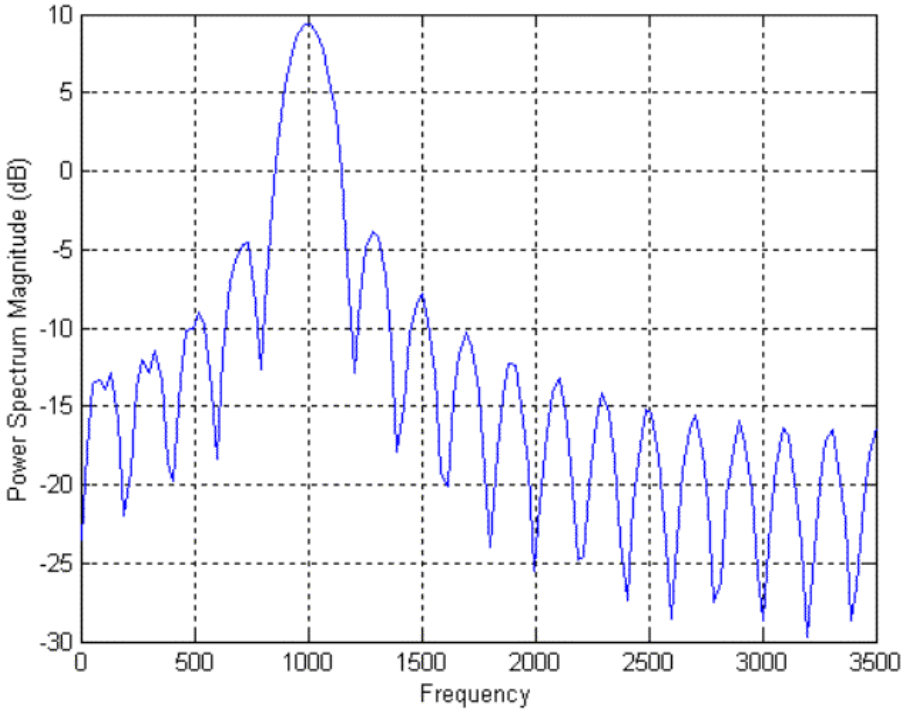
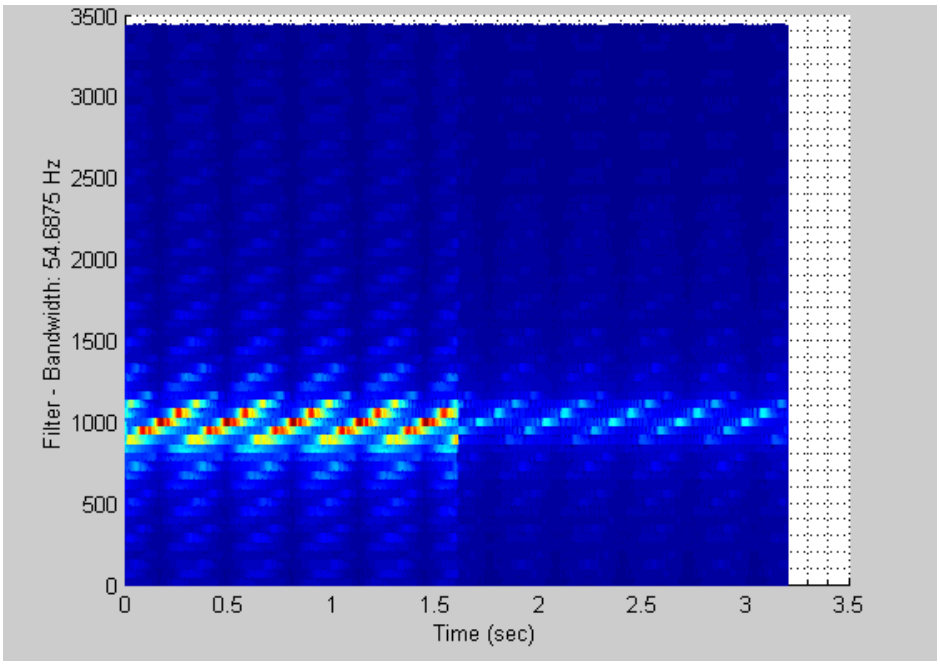


Figure 145 P4 Code N=64 cycles per phase =5 signal only PSD.

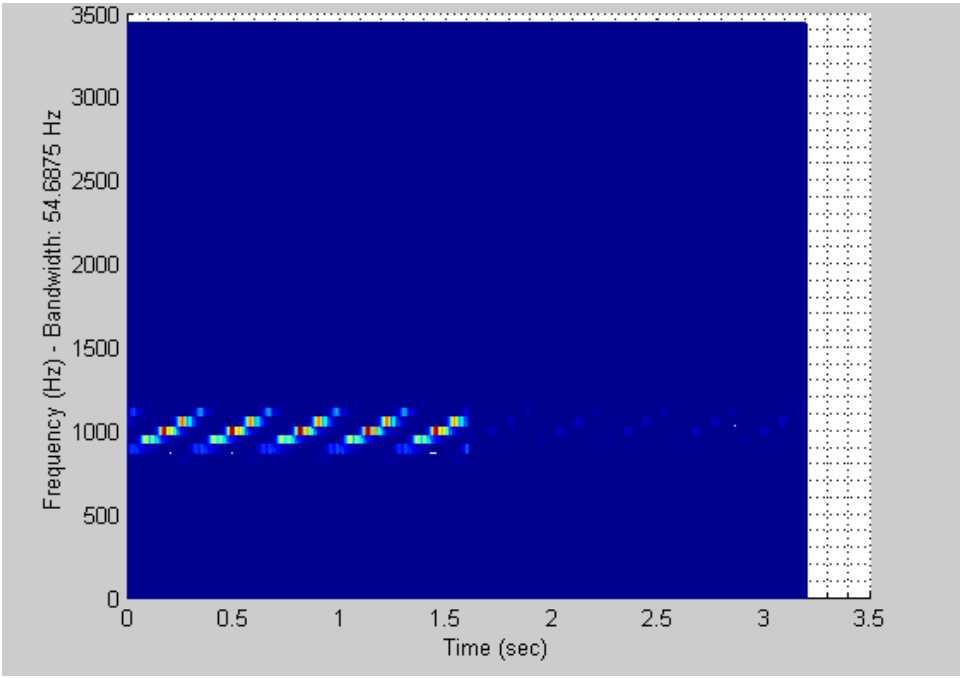
The response of the parallel filter arrays is illustrated in Figure 146 (a) .As in previous analysis; this filter bank separates the input signal in smaller sub-bands, providing a time-frequency description of the signal. Additionally, this figure shows the symmetry of the phase shifts at the center frequency.

A third-order cumulant estimator is applied to each one of the filters and the output signal is shown in Figure 146 (b). The resulting signal doesn't suffer a dramatic change due to the absence of noise added to this test signal. A zoom in this plot is presented in Figure 147 (a), where carrier frequency, bandwidth and code period are measured and compared with the input values in the previous table.

Moreover, Figure 147 (b) presents an amplitude-frequency plot confirming some of the previous measurements.

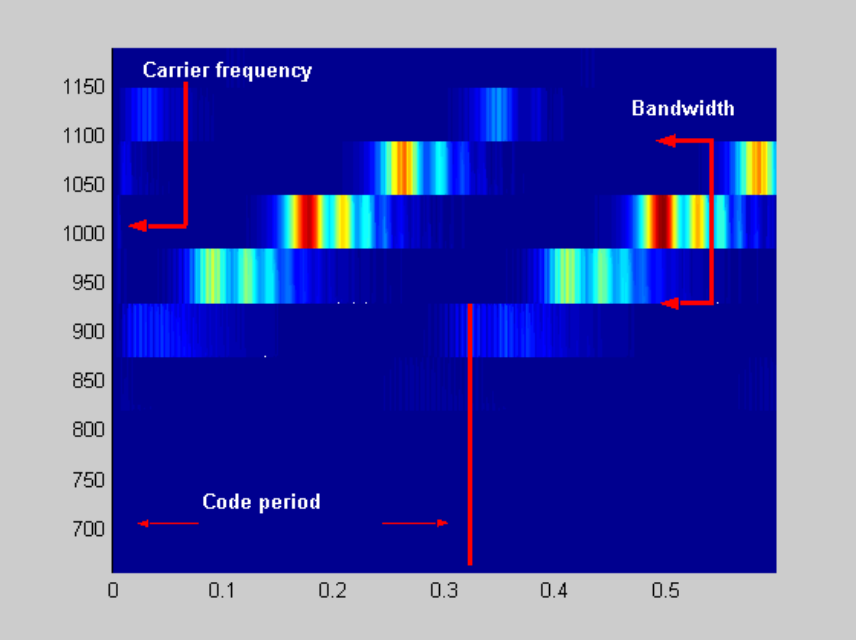


(a)

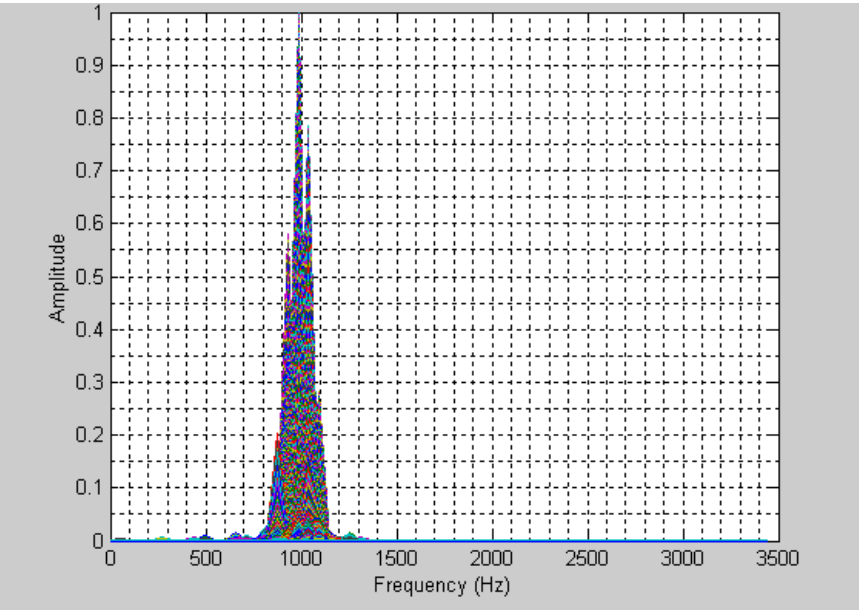


(b)

Figure 146 P4 Code N=64 cycles per phase =5 signal only (a) Output of the parallel filter arrays (b) Output after HOS.



(a)



(b)

Figure 147 P4 Code N=64 cycles per phase =5 signal only (a) Zoom in the resulting signal after HOS (b) Amplitude-frequency plot.

2. P4 Code, N=64, cycles per phase =5 and SNR=0 dB

Table 34 shows a P4-coded signal with carrier frequency equal to 1000 Hz, sampling frequency equal to 7000 Hz, 64 phases, 5 cycles per phase and SNR = 0 dB. Code period and bandwidth are calculated by the Equations (4.1.22) and (4.1.23). Figure 148 shows the PSD of the input signal, illustrating the distribution of the power along the different frequencies contained in the signal.

P4 – Parameters	Input Signal	Obtained	Comment
Carrier frequency (Hz)	1000	1000	
Sampling frequency (Hz)	7000	7000	Given
N – Phases	64	-	
SNR (dB)	0	-	
Cycles per phase	5	5	
Bandwidth	200	218.75	
Code Period (ms)	320	320	

Table 34 P4 Code N=64 cycles per phase =5 SNR=0 dB.

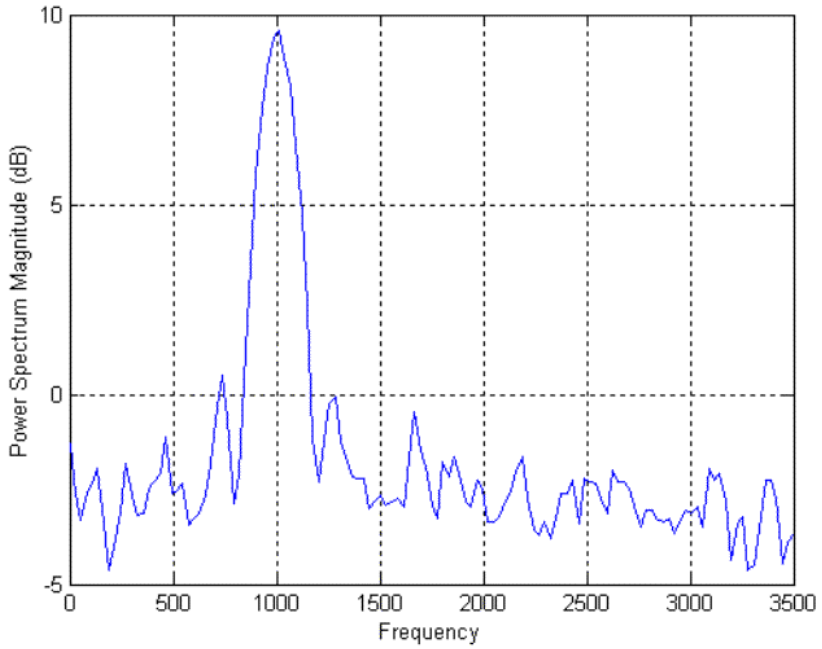


Figure 148 P4 Code N=64 cycles per phase =5 SNR=0 dB PSD.

The response of the parallel filter arrays for the P4-coded signal described above is presented in Figure 149 . Although SNR= 0 dB, this filter bank provides a clear frequency-time description of the input signal. Once the signal is separated into small sub-band signals, a third-order cumulant estimator is applied to each one of the sub-band signals. Figure 150 (a) illustrates the resulting signal after HOS, where the white Gaussian noise was suppressed. Because of some inter-modulation products, low-frequency signals become visible in the plots. With SNR = 0 dB, the introduced noise doesn't interfere in the detection problem.

Figure 150 (b) shows a zoom in the resulting signal after HOS. Carrier frequency, bandwidth and code period are measured and compared in the previous table. Figure 151 provides amplitude-filter and amplitude-frequency views of the resulting signal.

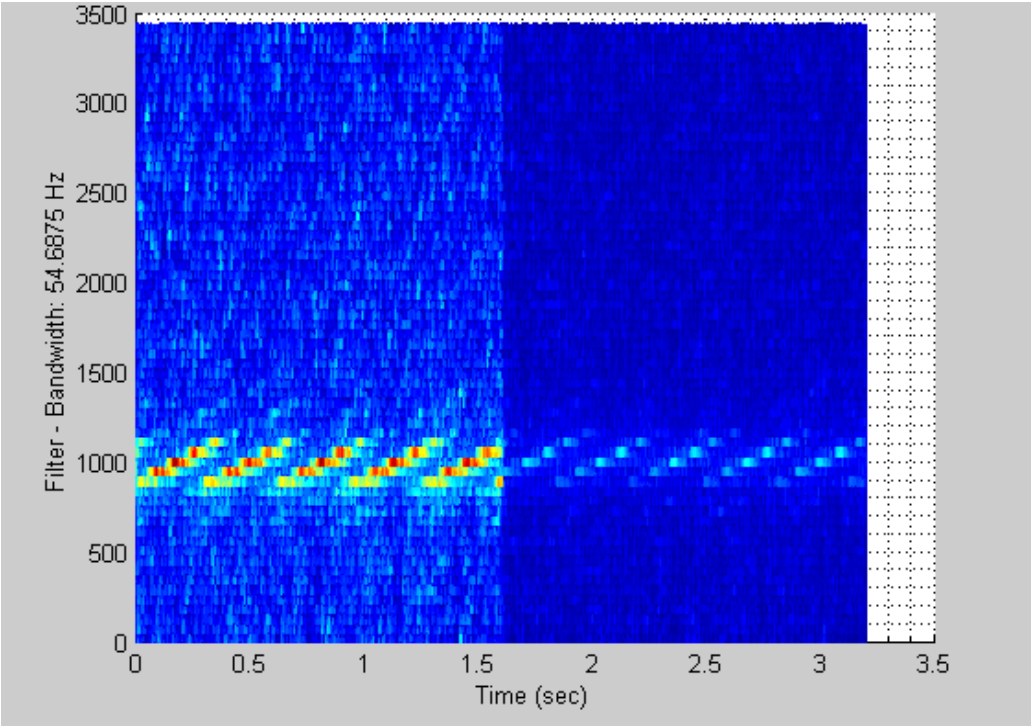
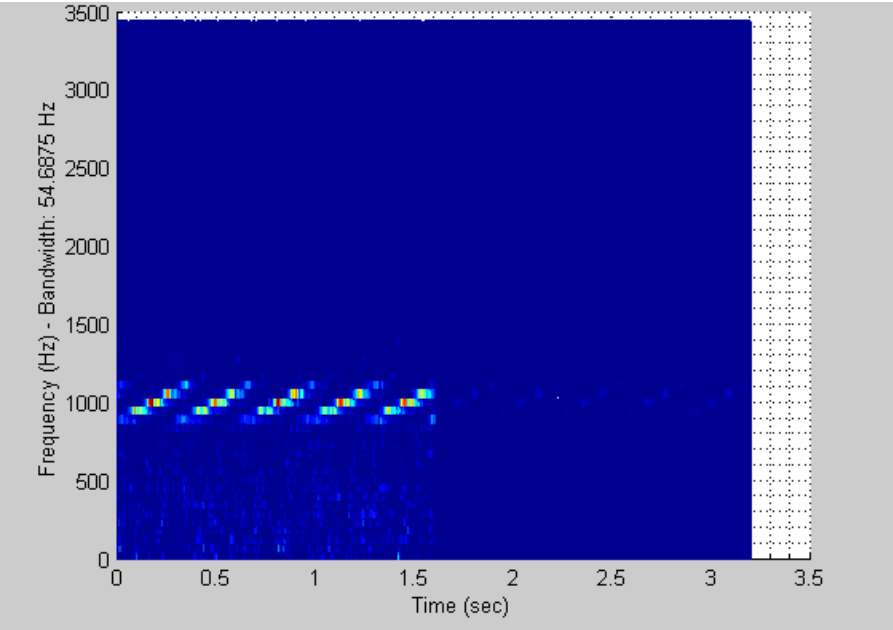
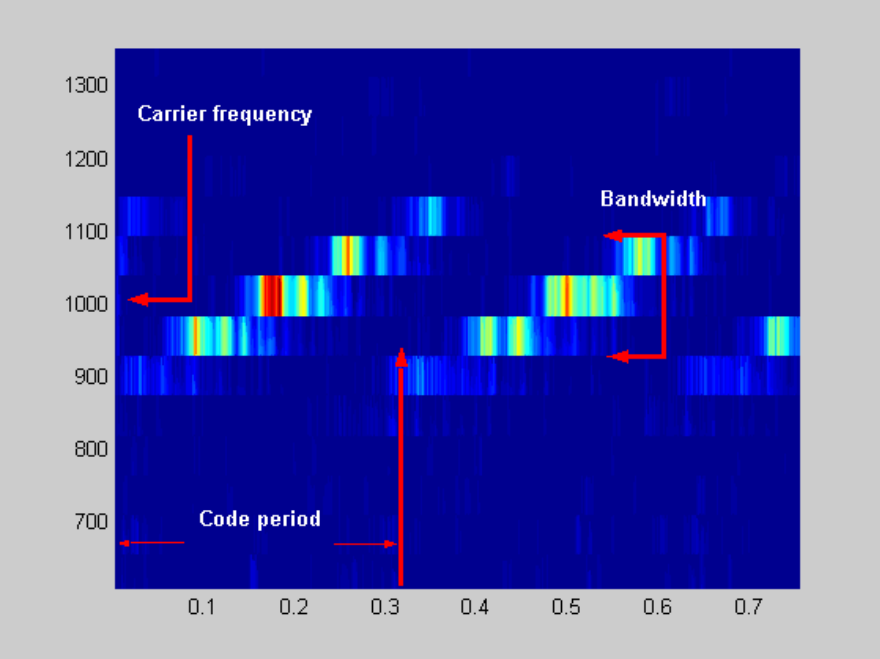


Figure 149 P4 Code N=64 cycles per phase =5 SNR=0 dB output of the parallel filter arrays.

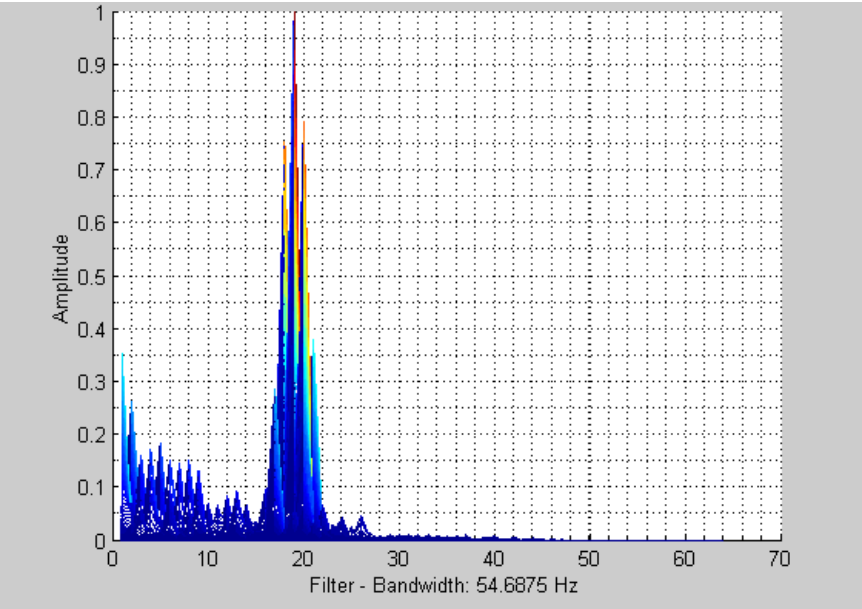


(a)

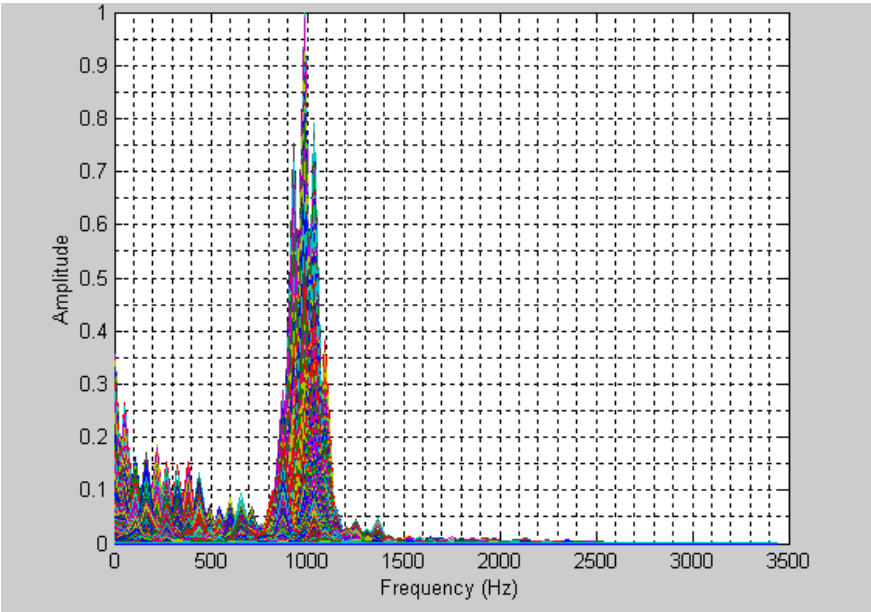


(b)

Figure 150 P4 Code N=64 cycles per phase =5 SNR=0 dB (a) Plot after HOS (b) Zoom in the resulting plot after HOS.



(a)



(b)

Figure 151 P4 Code N=64 cycles per phase =5 SNR=0 dB (a) Amplitude-filter plot (b) Amplitude-frequency plot.

.

3. Summary

Figure 152 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth, code period and number of phases in P4-coded signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very accurate values for carrier frequency, bandwidth and code period in any environment analyzed with a lower performance when SNR is less than –6 dB. The number of phases can be only extracted when noise is not presented and a number of cycles per phase is greater than 5.

P4	Carrier freq.(Hz)	Bandwidth (Hz)	Code period (ms)	Phases
Signal only	100%	100%	100%	100%
0 dB	100%	100%	100%	0%
(-) 6 dB	100%	103%	97%	0%

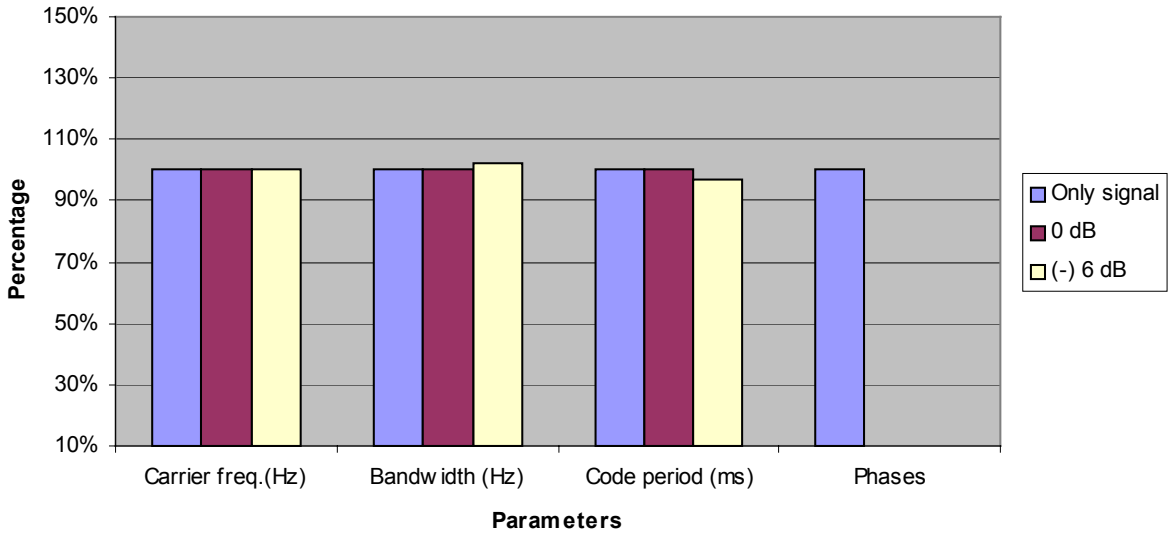


Figure 152 Performance of the signal processing detecting P4-coded signals.

I. COSTAS CODE

Chapter II describes the characteristics of LPI signals modulated by Costas Code. In this frequency hopping system, the signal consists of one or more frequencies being chosen from a set $\{f_1, f_2, ..., f_m\}$ of available frequencies, for transmission at each of a set $\{t_1, t_2, ..., t_n\}$ of consecutives time intervals. For this simulation, a set of twelve different Costas-coded signals was generated considering the situation in which $m=n$, and a different one of n equally spaced frequencies $\{f_1, f_2, ..., f_n\}$ is transmitted during each of the n equal duration time intervals $\{t_1, t_2, ..., t_n\}$. Table 35 show the matrix of Costas-coded signals were two different Costas sequences were used. Additionally, cycles per phase and SNR were varied to evaluate the performance of the proposed signal processing. Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	File	Carrier Frequency (Hz)	Sampling Frequency (Hz)	Transmission time in each frequency (ms)	SNR (dB)
1	C_1_15_10_s	1000	15000	10	-
2	C_1_15_10_0	1000	15000	10	0
3	C_1_15_10_-6	1000	15000	10	-6
4	C_1_15_20_s	1000	15000	20	-
5	C_1_15_20_0	1000	15000	20	0
6	C_1_15_20_-6	1000	15000	20	-6
7	C_2_15_10_s	1000	15000	10	-
8	C_2_15_10_0	1000	15000	10	0
9	C_2_15_10_-6	1000	15000	10	-6
10	C_2_15_20_s	1000	15000	20	-
11	C_2_15_20_0	1000	15000	20	0
12	C_2_15_20_-6	1000	15000	20	-6

Table 35 Matrix of test signals for Costas code.

1. Costas code, sequence 1, time in frequency 10 ms, signal only

Table 36 describes a Costas-coded signal with a Costas sequence of 7 frequencies, sampling frequency equal to 15000 Hz and transmission time in each frequency equal to 10 ms. The PSD of the signal is presented in Figure 153 This figure shows the distribution of the power in the frequencies related to the Costas sequence 4-7-1-6-5-2-3, where each of this numbers represents frequencies in KHz. Additionally; the figure illustrates the total bandwidth of the signal, which is about 6000 Hz.

Costas– Parameters	Input Signal	Obtained	Comment
Costas Sequence (KHz)	4-7-1-6-5-2-3	4-7-1-6-5-2-3	
Sampling frequency (Hz)	15000	15000	Given
SNR (dB)	Signal only	-	
Transmission time per frequency (ms)	10	10	
Code period (ms)	70	70	
Bandwidth (Hz)	6000	6014.8	

Table 36 Costas code, sequence 1, time in frequency 10 ms, signal only.

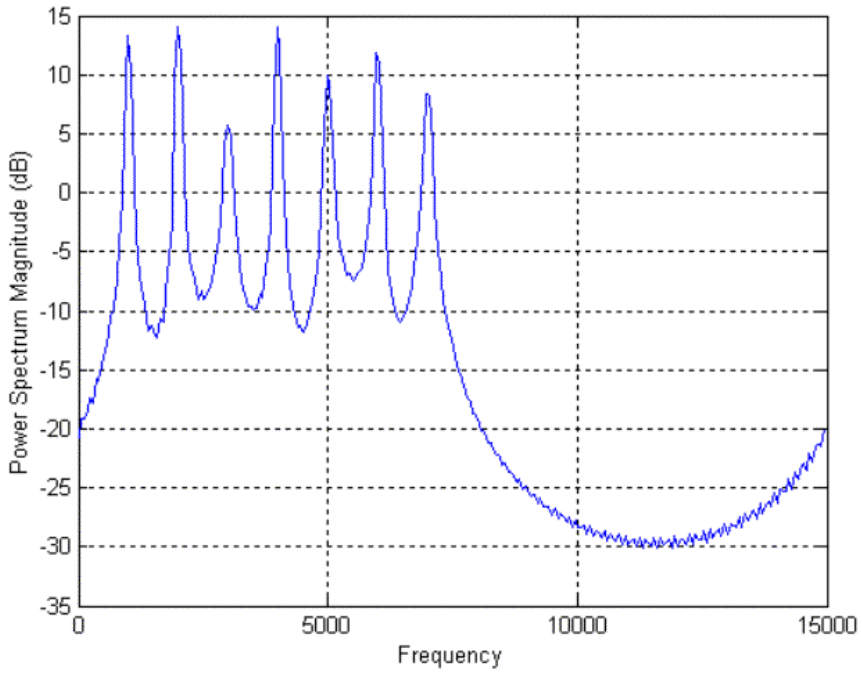


Figure 153 Costas code, sequence 1, time in frequency 10 ms, signal only PSD.

Figure 154 shows the output signal after the parallel filter arrays. This plot presents a frequency-time description of the signal where the Costas sequence of frequencies can be identified. The bandwidth of each sub-filter is directly proportional to the sampling frequency f_s and inversely proportional to twice the number of filters L in the array

$$\text{Bandwidth} = \frac{f_s}{2L} = \frac{15000}{2*64} = 117.68$$

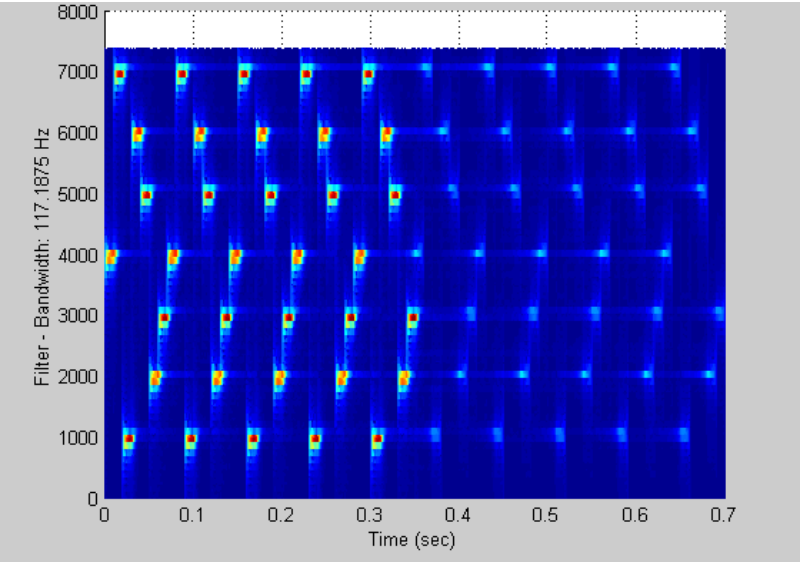
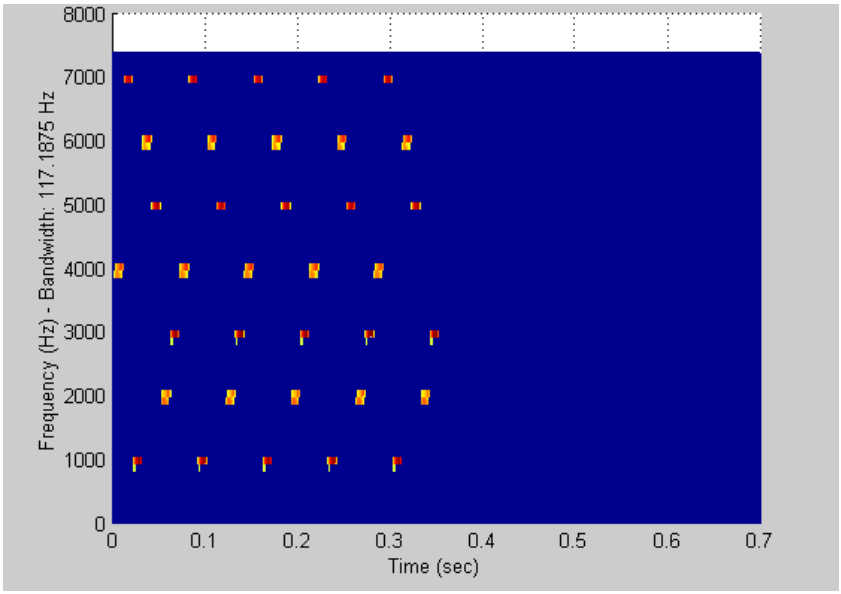


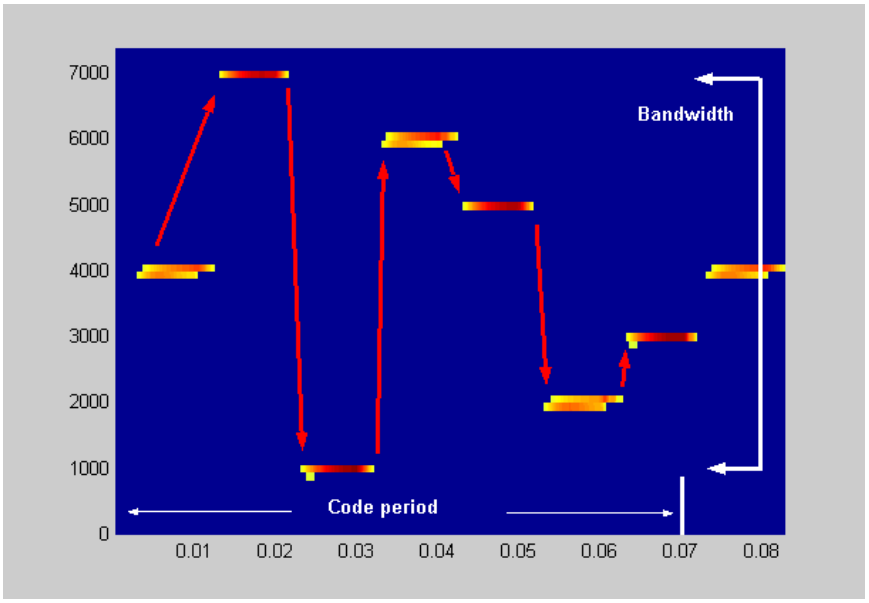
Figure 154 Costas code, sequence 1, time in frequency 10 ms, signal only Output of the parallel filter arrays.

Figure 155 (a) illustrates the output signal after third-order cumulant estimators are applied to each one of the sub-filters. Noise is not added to the signal; therefore, the result of the HOS is just a degradation of the previous signal. Figure 155 (b) presents a zoom in the output after HOS. The Costas sequence, bandwidth and code period are measured and compared in the previous table.

In addition, Figure 156 provides an amplitude-frequency view of the resulting signal after HOS. All the frequency in the Costas code and the total bandwidth is confirmed with the obtained data.



(a)



(b)

Figure 155 Costas code, sequence 1, time in frequency 10 ms, signal only (a) Output after HOS (b) Zoom in plot after HOS showing parameters.

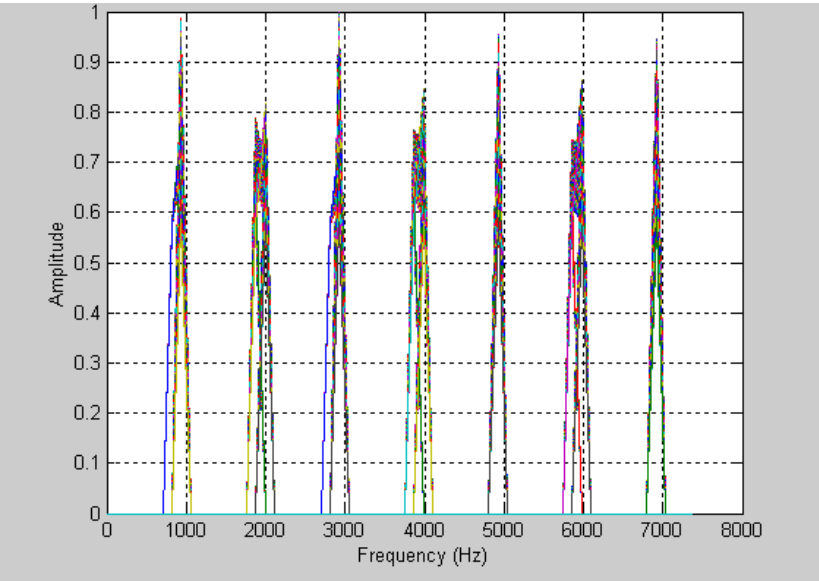


Figure 156 Costas code, sequence 1, time in frequency 10 ms, signal only amplitude-frequency plot

2. Costas code, sequence 1, time in frequency 10 ms, SNR=0 dB

Table 37 describes a Costas-coded signal with a Costas sequence of 7 frequencies, sampling frequency equal to 15000 Hz, transmission time in each frequency equal to 10 ms and SNR = 0 dB. The PSD of the signal is presented in Figure 157 .This figure shows the distribution of the power in the frequencies related to the Costas sequence 4-7-1-6-5-2-3, where each of this numbers represents frequencies in KHz. In addition; the figure illustrates the total bandwidth of the signal, which is about 6000 Hz.

Costas– Parameters	Input Signal	Obtained	Comment
Costas Sequence (KHz)	4-7-1-6-5-2-3	4-7-1-6-5-2-3	
Sampling frequency (Hz)	15000	15000	Given
SNR (dB)	0	-	
Transmission time per frequency (ms)	10	10	
Code period (ms)	70	70	
Bandwidth (Hz)	6000	6014.8	

Table 37 Costas code, sequence 1, time in frequency 10 ms, SNR=0 dB.

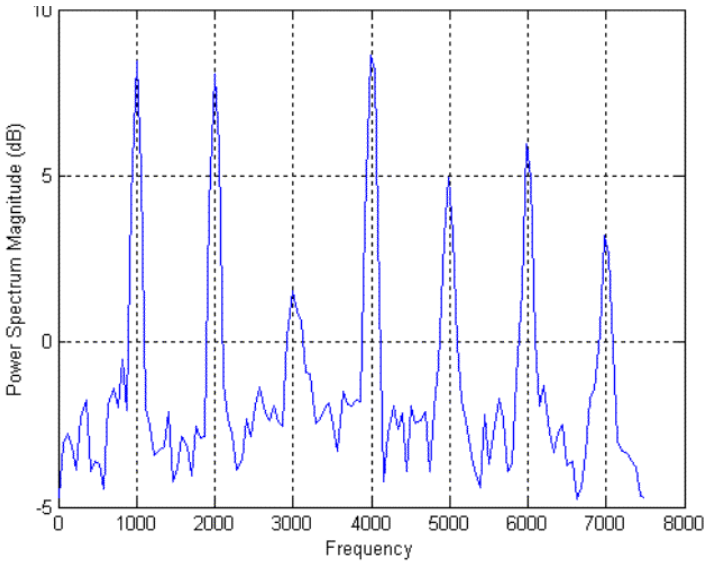


Figure 157 Costas code, sequence 1, time in frequency 10 ms, SNR=0 dB PSD.

Figure 158 shows the response of the filter bank. The input signal was decomposed into small sub-band signals to provide a frequency-time representation. The bandwidth of each sub-filter is kept constant through the evaluation of Costas-coded signal as a result of keeping the number of filters and sampling frequency constant.

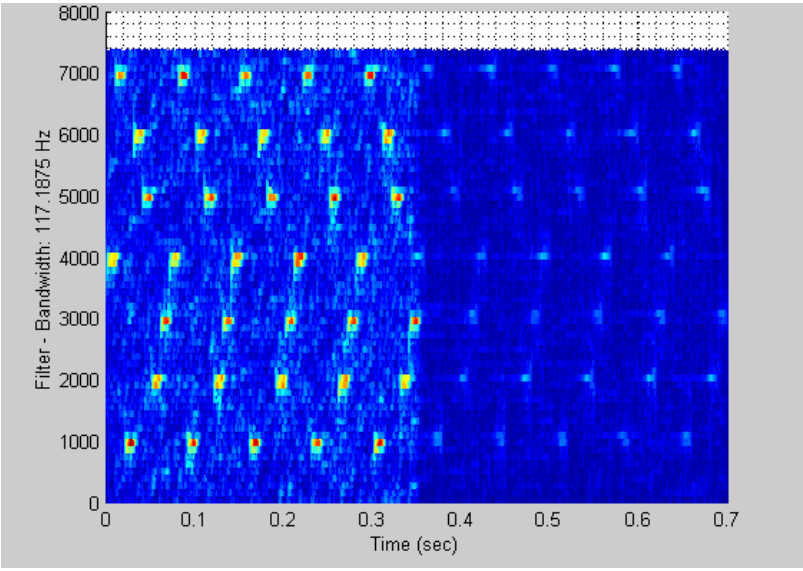
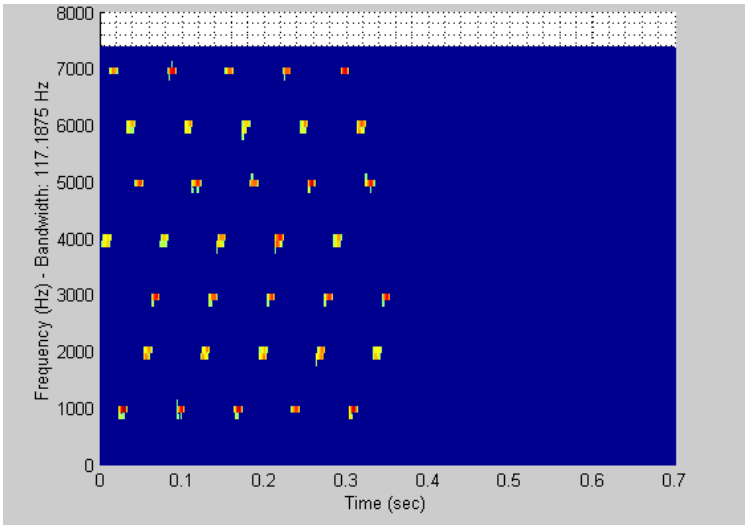
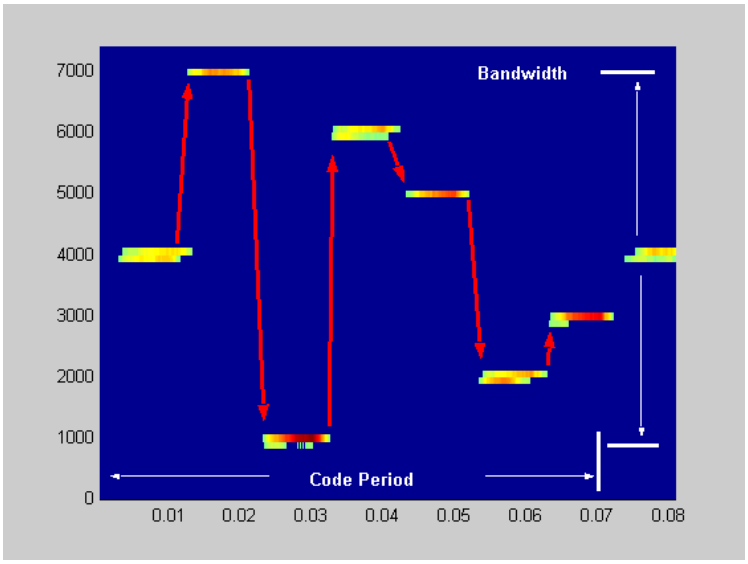


Figure 158 Costas code, sequence 1, time in frequency 10 ms, SNR=0 dB Output of the parallel filter arrays.

Figure 159 (a) shows the output signal after applying third-order cumulant estimators to each of the sub-filters. Even though the SNR = 0 dB, all the characteristics of the signal can be extracted form the plot. Figure 159 (b) provides a zoom in the previous plot where all the parameters are measured and compared in the previous table. An amplitude-frequency plot of the resulting signal is provided in Figure 160 .



(a)



(b)

Figure 159 Costas code, sequence 1, time in frequency 10 ms, SNR=0 dB (a) Output after HOS (b) Zoom in the resulting signal after HOS.

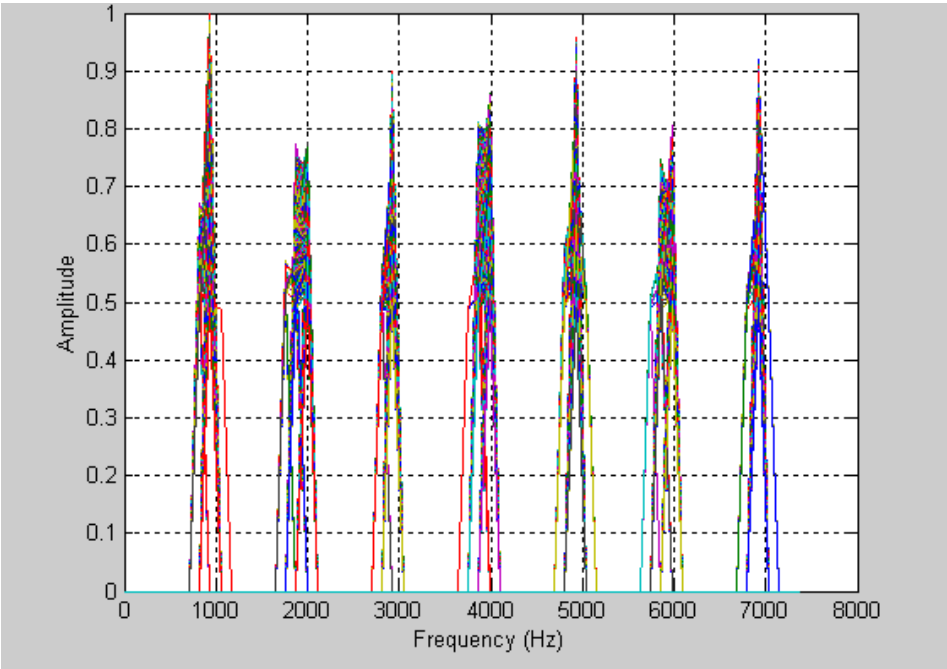


Figure 160 Costas code, sequence 1, time in frequency 10 ms, SNR=0 dB amplitude-frequency plot.

3. Summary

Figure 161 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth, and code period in Costas-coded signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very accurate values for carrier frequency, bandwidth and code period in any environment analyzed.

COSTAS	Sequence	time in frequency(ms)	Code period (ms)
Signal only	100%	100%	100%
0 dB	100%	100%	100%
(-) 6 dB	100%	100%	100%

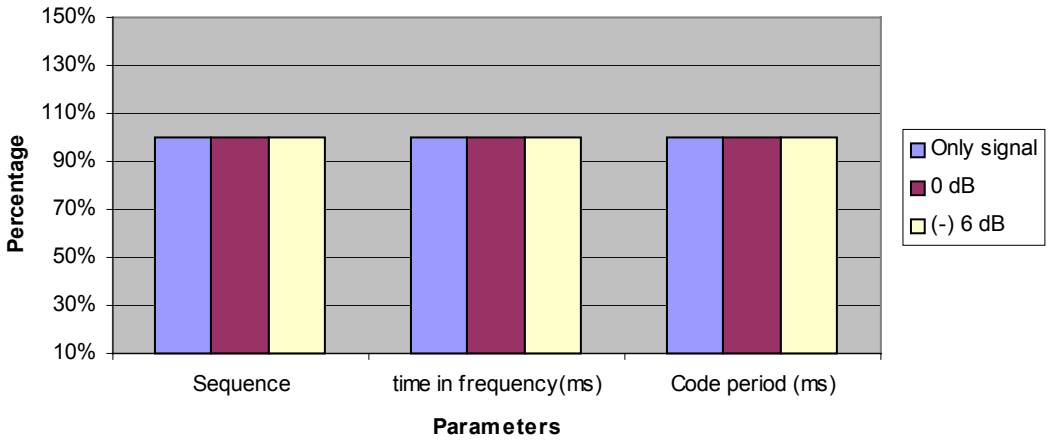


Figure 161 Performance of the signal processing detecting Costas-coded signals.

J. FSK/PSK COMBINED WITH COSTAS CODE

As mentioned in Chapter II, this modulation technique is the result of a combination of frequency shift keying based on a Costas frequency hopping matrix and phase shift keying using Barker sequences of different lengths. In a Costas frequency hopped signal, the firing order defines what frequencies will appear and with what duration. During each sub-period, as the signal stays in one of the frequencies, a binary phase modulation occurs according to a Barker. The final waveform may be seen as a binary phase shifting modulation within each frequency hop, resulting in 25 phase slots equally distributed in each frequency.

With the purpose of testing the signal processing based on the use of parallel filter arrays and HOS, eight signals are generated and analyzed (Table 38). This section describes the input signal in detail through the PSD, a time domain representation and the PAF. Then, the input signal is evaluated with the proposed signal processing. The input parameters and results are compared in tables. Signals with SNR= -6 dB are not analyzed in this section because of the proposed method is unable to detect and identify these particular signals. Only the analysis of one signal is presented in this thesis. The rest of the results are included in a technical report to be published.

No	File	F_s (Hz)	No of Barker Bits	Cycles per bit	SNR (dB)
1	FSK_PSK_C_1_15_5_1_s	15000	5	1	-
2	FSK_PSK_C_1_15_5_1_0	15000	5	1	0
3	FSK_PSK_C_1_15_5_5_s	15000	5	5	-
4	FSK_PSK_C_1_15_5_5_0	15000	5	5	0
5	FSK_PSK_C_1_15_11_1_s	15000	11	1	-
6	FSK_PSK_C_1_15_11_1_0	15000	11	1	0
7	FSK_PSK_C_1_15_11_5_s	15000	11	5	-
8	FSK_PSK_C_1_15_11_5_0	15000	11	5	0

Table 38 Matrix of test signals for FSK/PSK Costas code.

1. FSK/PSK costas, bits in code =5, cycle per bit =1, signal only

Table 39 describes a FSK/PSK Costas-coded signal with a Costas sequence of 7 frequencies, sampling frequency equal to 15000 Hz, 5- bit Barker code and 1 cycles per bit. The PSD of the signal is presented in Figure 162 . This figure shows the distribution of the power in the frequency domain; in addition, the figure illustrates the total bandwidth of the signal.

FSK/PSK Costas– Parameters	Input Signal	Obtained	Comment
Costas Sequence (KHz)	4-7-1-6-5-2-3	4-7-1-6-5-2-3	
Sampling frequency (Hz)	15000	15000	Given
Number of bits per Barker code	5	5	
Cycles per Barker bit	1	1	
Bandwidth (Hz)	7000	7053	

Table 39 FSK/PSK costas, bits in code =5, cycle per bit =1, signal only.

The bandwidth in each one of the carry frequencies used is kept constant; therefore, the number of cycle per bit must vary as shown in Equations 4.1.28 and 4.1.30. As a consequence of keeping the same bandwidth for each frequency, the code period and the time spent by the signal in each frequency is constant as demonstrated in 4.1.29 and 4.1.31. For example the bandwidth for the signal with carrier frequency 1 KHz is

$$B_{1KHz} = \frac{f_c}{\text{Cycles per phase}} = \frac{1000 \text{ Hz}}{1 \text{ cycles per phase}} = 1000 \text{ Hz} \tag{4.1.24}$$

and the code period is

$$t_c = \frac{(\text{Cycles/bit})(\text{No of bits})}{f_c} = \frac{(1)(5)}{1000} = 5 \text{ ms} \tag{4.1.25}$$

The bandwidth for the signal with carrier frequency 7 KHz is

$$B_{7KHz} = \frac{f_c}{\text{Cycles per phase}} = \frac{7000 \text{ Hz}}{7 \text{ cycles per phase}} = 1000 \text{ Hz} \tag{4.1.26}$$

and the code period is

$$t_c = \frac{(\text{Cycles/bit})(\text{No of bits})}{f_c} = \frac{(7)(5)}{7000} = 5 \text{ ms} \tag{4.1.27}$$

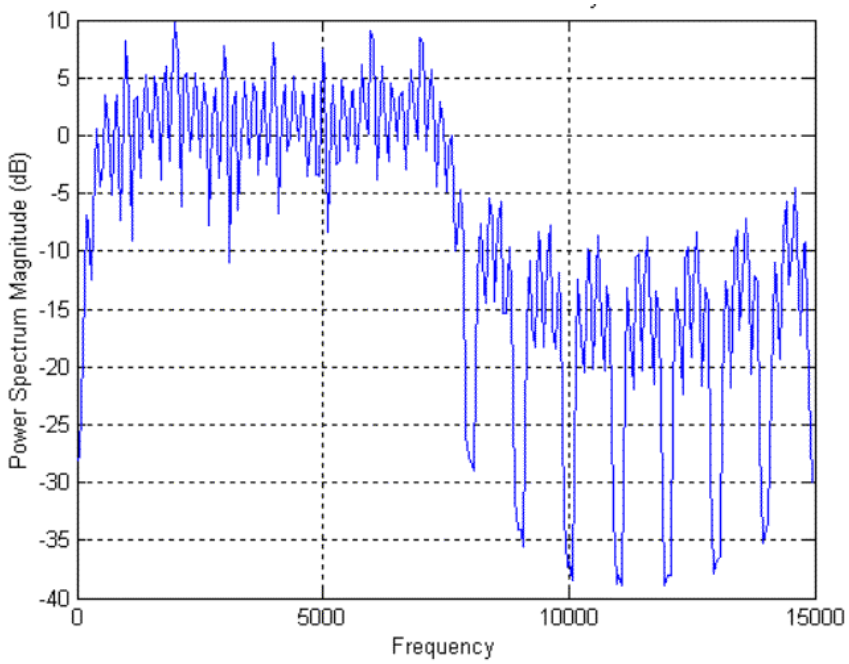


Figure 162 FSK/PSK Costas, bits in code =5, cycle per bit =1, signal only PSD.

The response of the parallel filter arrays is presented in Figure 163 . The input signal is decomposed into small sub-band signals. The resolution of the filter bank depends on the bandwidth of each sub-filter. The bandwidth of each sub-filter is calculated by $f_s/(2L)$ where f_s represents the sampling frequency and L the number of filters in the array. In this case the minimum resolution of the array is 117.18 Hz. This figure provides a good time-frequency representation where a general behavior of the signal can be identified.

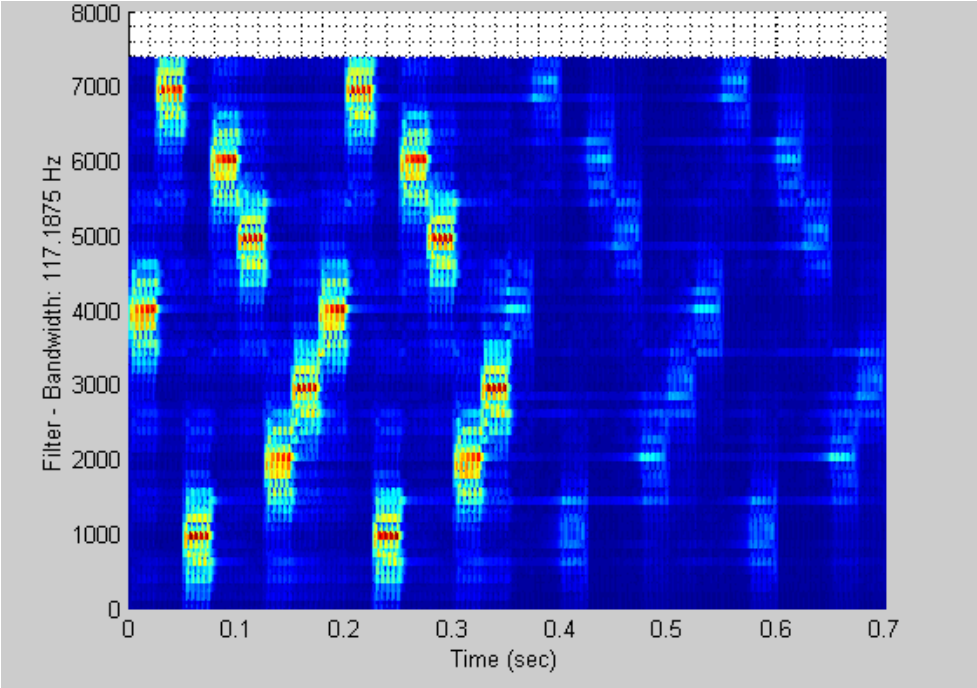


Figure 163 FSK/PSK Costas, bits in code =5, cycle per bit =1, signal only output of the parallel filter arrays.

Figure 164 shows the resulting signal after a third-order cumulant estimator is applied to each one of the sub-band signals after the parallel filter arrays. Due to the signal lacks of noise, the resulting signal is not very different from the previous signal. An important improvement will be noticed when the noise is many time greater than the signal.

Figure 165 zooms in the previous plot and shows the Costas sequence followed by the signal during one period. Figure 166 reveals many characteristics of the signal. Carrier frequency, bandwidth of the signal for each frequency, code period and the time spent by the signal in each frequency is recorded and compared with the input values.

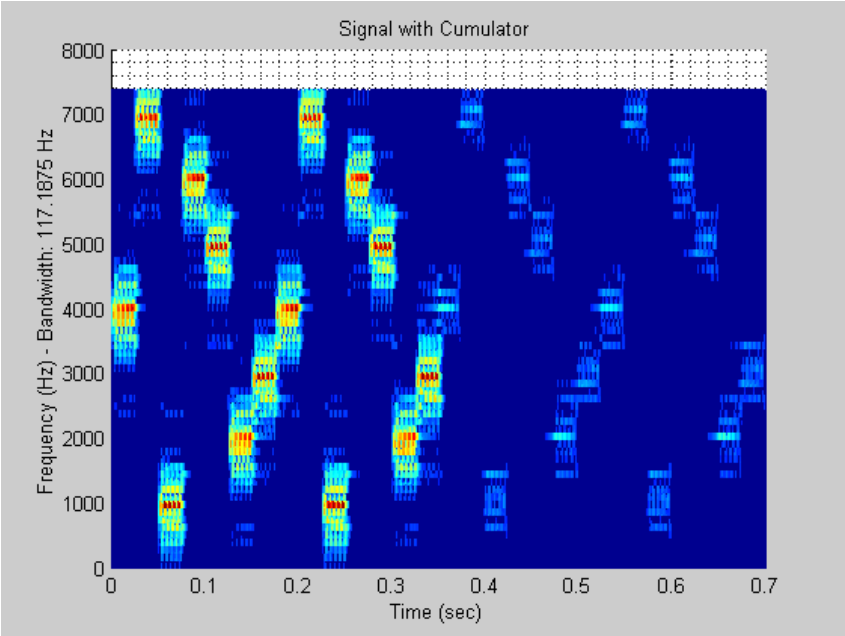


Figure 164 FSK/PSK Costas, bits in code =5, cycle per bit =1, signal only output after HOS.

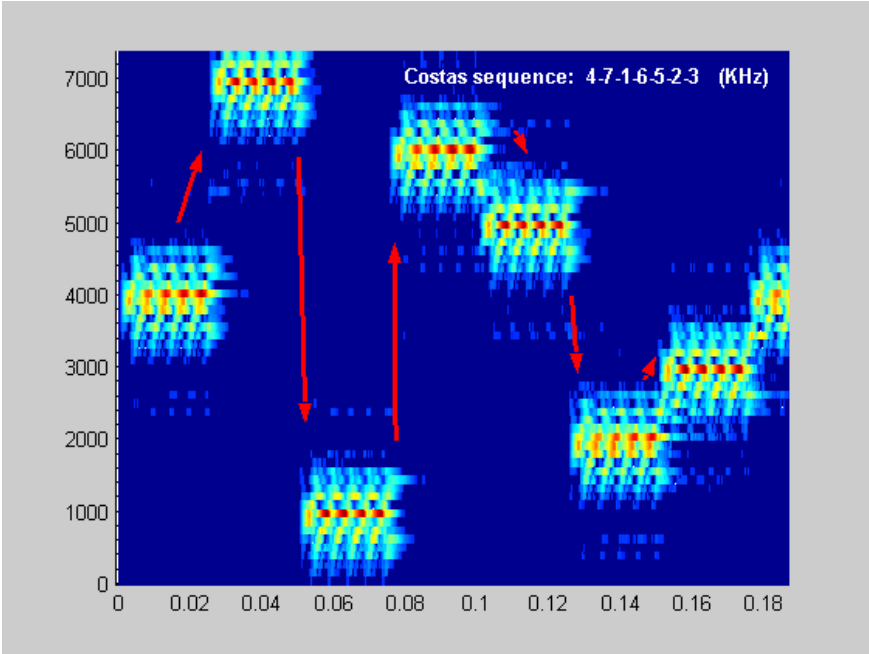


Figure 165 FSK/PSK Costas, bits in code =5, cycle per bit =1, signal only Zoom in the previous figure showing the Costas sequence.

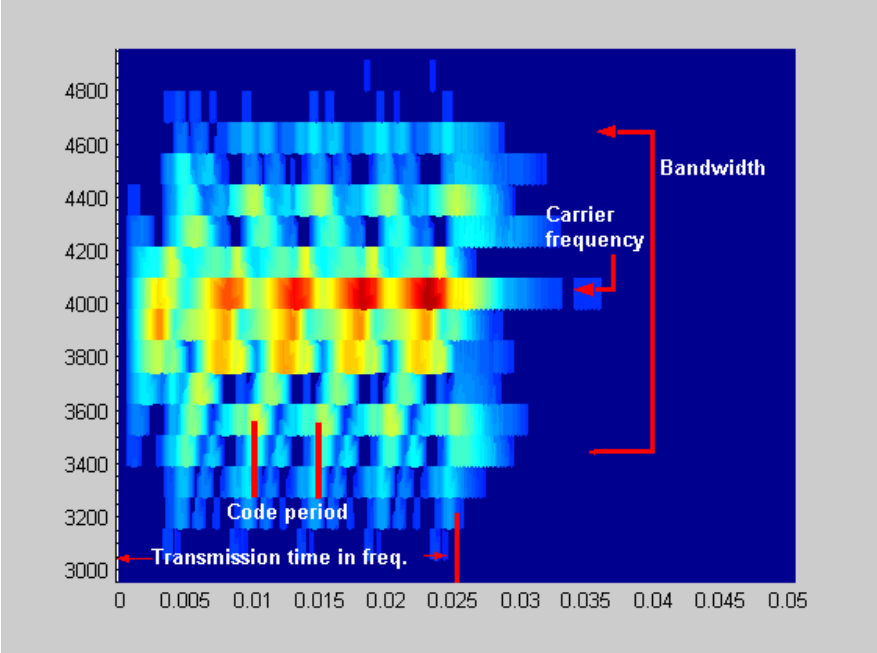


Figure 166 FSK/PSK Costas, bits in code =5, cycle per bit =1, signal only Zoom in the previous figure showing parameters.

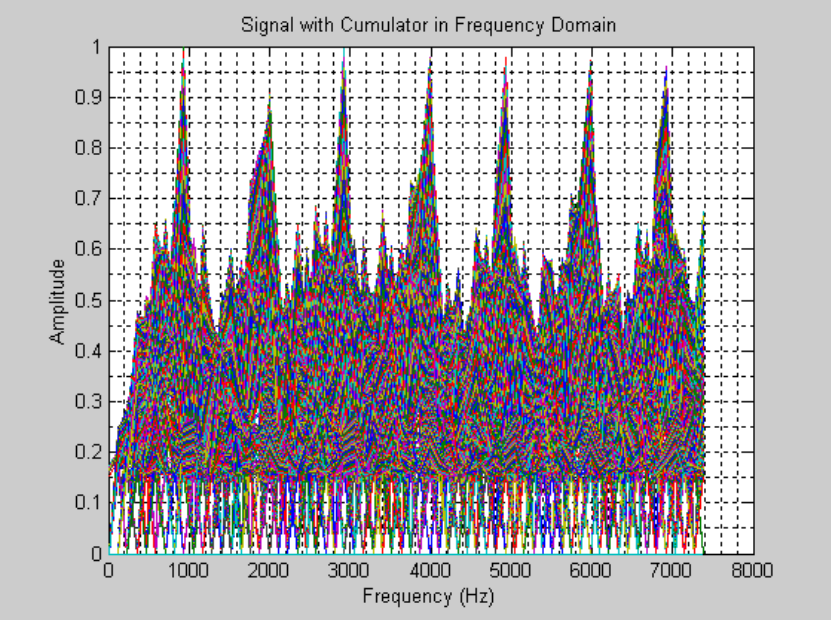


Figure 167 FSK/PSK Costas, bits in code =5, cycle per bit =1, signal only amplitude-frequency plot of the resulting signal.

2. FSK/PSK Costas, bits in code =5, cycle per bit =1, SNR=0 dB

Table 40 describes a FSK/PSK Costas-coded signal with a Costas sequence of 7 frequencies, sampling frequency equal to 15000 Hz, 5-bit Barker code, 1 cycles per bit and SNR= 0 dB. The PSD of the signal is presented in Figure 168 . This figure shows the distribution of the power in the frequencies; in addition, the figure illustrates the total bandwidth of the signal.

FSK/PSK Costas– Parameters	Input Signal	Obtained	Comment
Costas Sequence (KHz)	4-7-1-6-5-2-3	4-7-1-6-5-2-3	
Sampling frequency (Hz)	15000	15000	Given
Number of bits per Barker code	5	5	
Cycles per Barker bit	1	1	
Bandwidth (Hz)	7000	7053	
SNR (dB)	0	0	

Table 40 FSK/PSK Costas, bits in code =5, cycle per bit =1, SNR=0 dB.

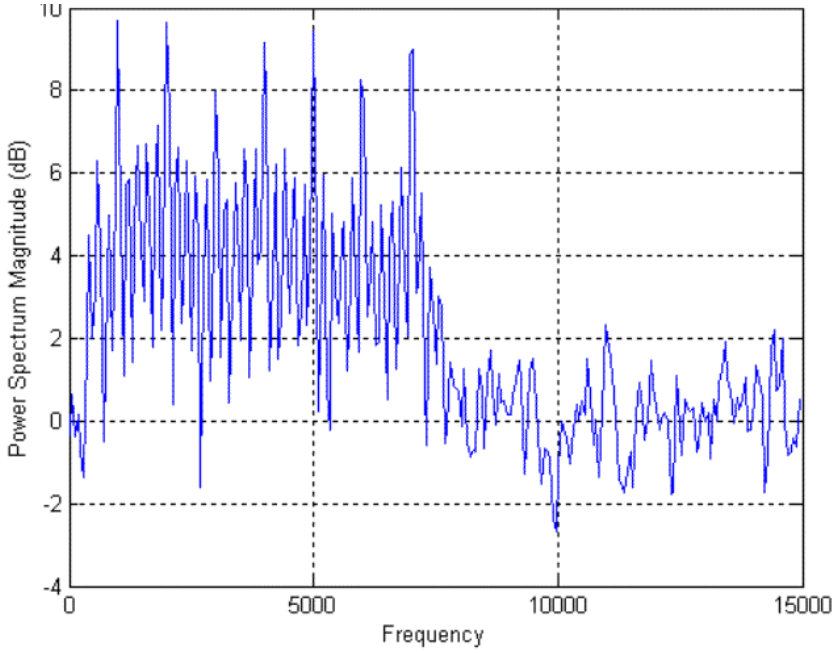


Figure 168 FSK/PSK Costas, bits in code =5, cycle per bit =1, SNR=0 dB PSD.

The response of the parallel filter arrays is presented in Figure 169 . The input signal is decomposed into small sub-band signals. This figure provides a good time-frequency representation where a general behavior of the signal can be identified.

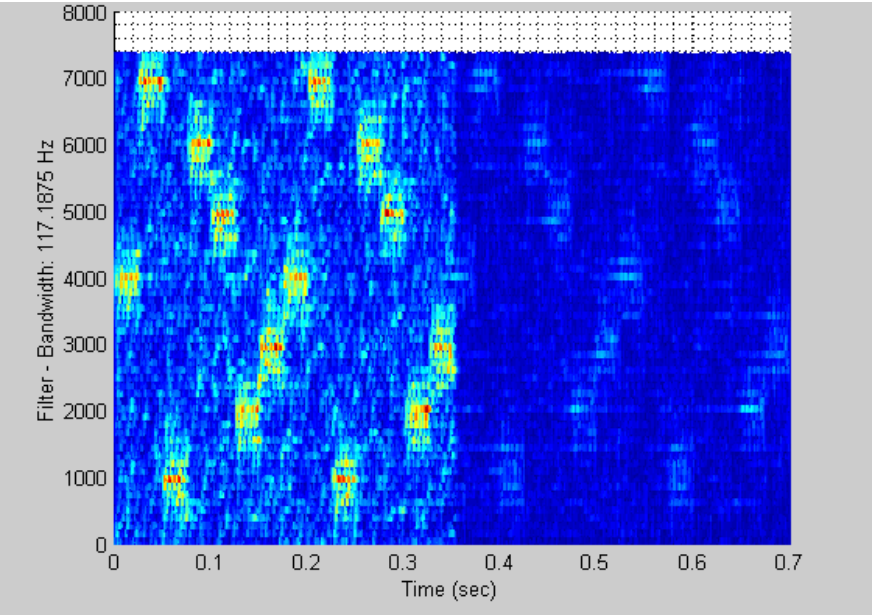


Figure 169 FSK/PSK Costas, bits in code =5, cycle per bit =1, SNR=0 dB Output of the parallel filter bank.

Figure 170 shows the resulting signal after a third-order cumulant estimator is applied to each one of the sub-band signals after the parallel filter arrays. Figure 171 zooms in the previous plot and shows the Costas sequence followed by the signal during one period. Figure 172 reveals many characteristics of the signal. Carrier frequency, bandwidth of the signal for each frequency, code period and the time spent by the signal in each frequency is recorded and compared with the input values. Figure 173 shows an amplitude-frequency plot of the resulting signal after HOS.

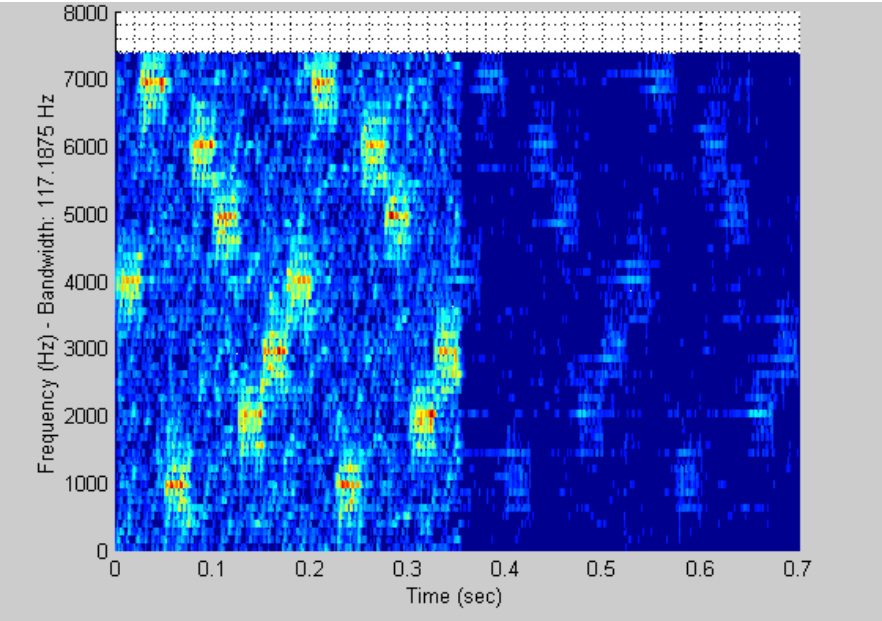


Figure 170 FSK/PSK Costas, bits in code =5, cycle per bit =1, SNR=0 dB output after HOS.

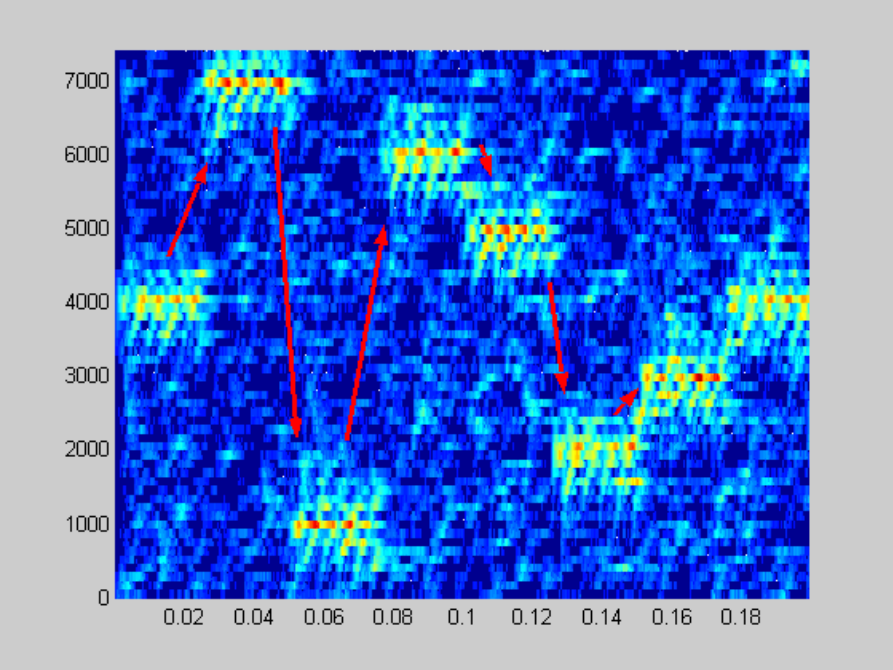


Figure 171 FSK/PSK Costas, bits in code =5, cycle per bit =1, SNR=0 dB zoom in the previous plot after HOS.

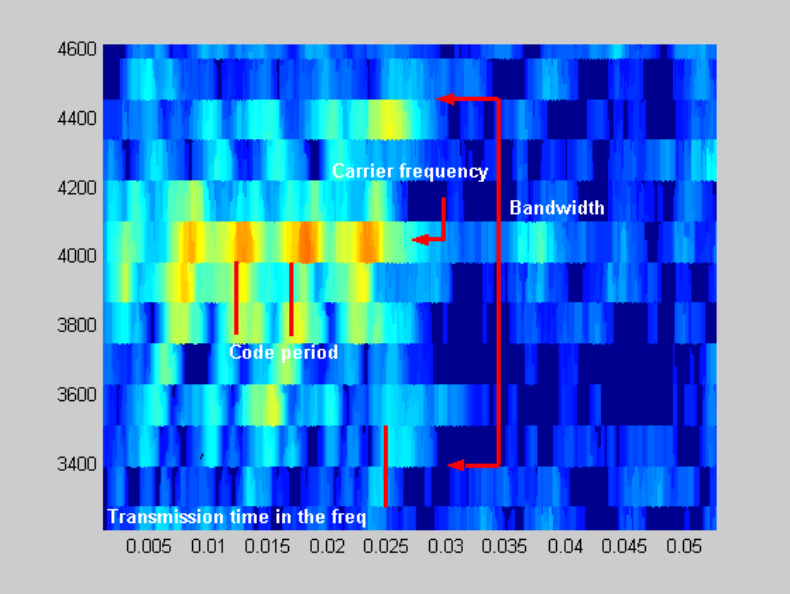


Figure 172 FSK/PSK Costas, bits in code =5, cycle per bit =1, SNR=0 dB zoom in previous plot showing parameters.

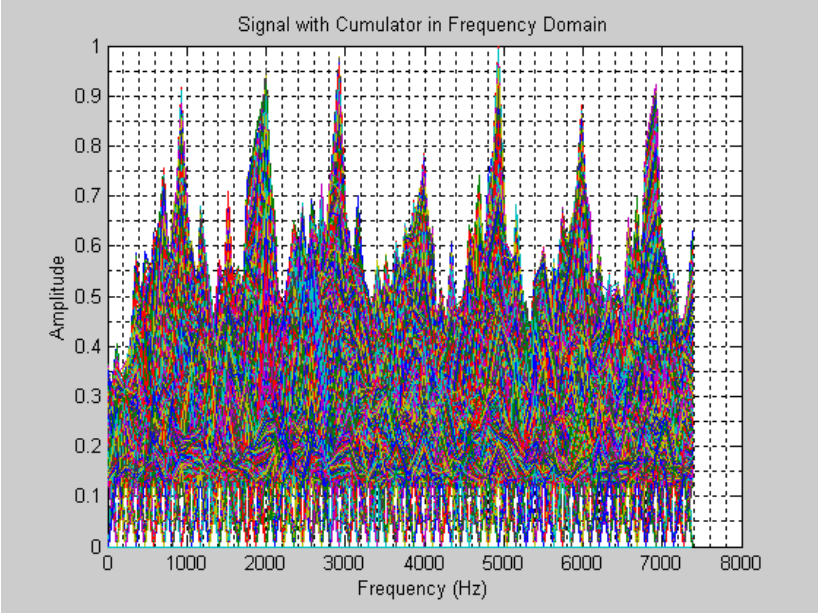


Figure 173 FSK/PSK Costas, bits in code =5, cycle per bit =1, SNR=0 dB Amplitude-frequency plot after HOS.

3. Summary

Figure 174 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth and code period in FSK/PSK Costas-coded signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide very precise values for carrier frequency, bandwidth and code period in any environment analyzed except for SNR lower than 0 dB where none of the parameters can be extracted.

FSK/PSK COSTAS	Sequence	Bandwidth (Hz)	Code period (ms)
Signal only	100%	100%	100%
0 dB	100%	100%	100%

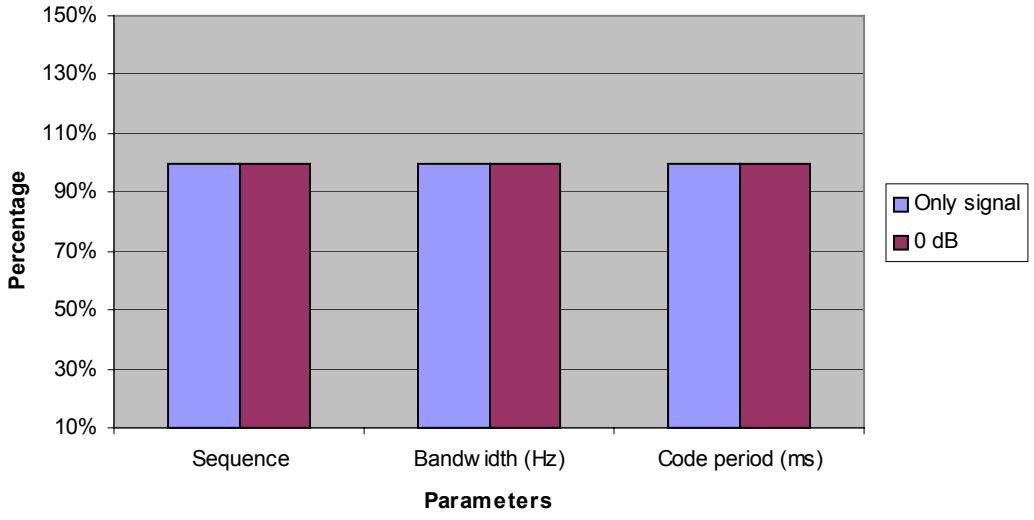


Figure 174 Performance of the signal processing detecting Costas-coded signals.

K. FSK/PSK TARGET

Chapter II describes the generation of FSK/PSK Target signals in detail. Instead of spreading the energy of the signal equally over a broad bandwidth, this type of technique concentrates the signal energy in specific spectral locations of most importance for the radar and its typical targets, within the broad-spectrum bandwidth. The produced signals have a pulse compression characteristic and therefore they can achieve LPI.

Eight signals are considered for the evaluation of the proposed signal processing detecting FSK/PSK Target signals. This thesis shows the analysis of one test signal. The analysis of the complete test signal matrix is included in the Technical Report. Signals with SNR= -6 dB are not analyzed in this section because of the method is unable to detect and identify these particular signals.

No	File	F_s (Hz)	No of phases	Cycles per phase	SNR (dB)
1	FSK_PSK_T_15_128_5_s	15000	128	5	-
2	FSK_PSK_T_15_128_5_0	15000	128	5	0
3	FSK_PSK_T_15_128_10_s	15000	128	10	-
4	FSK_PSK_T_15_128_10_0	15000	128	10	0
5	FSK_PSK_T_15_256_5_s	15000	256	5	-
6	FSK_PSK_T_15_256_5_0	15000	256	5	0
7	FSK_PSK_T_15_256_10_s	15000	256	10	-
8	FSK_PSK_T_15_256_10_0	15000	256	10	0

Table 41 Test signal matrix for FSK/PSK Target.

1. FSK/PSK Target, phases =128, cycle per phase =5, signal only

Table 42 describes a FSK/PSK target signal with sampling frequency 15 KHz, 5 different phases and 128 random frequencies. The PSD of the signal is shown in Figure 175 . This plot presents the distribution of the power along different frequency components of the signal. As a result of the target response there is a concentration of the power between 3 KHZ and 6 KHz. Within this interval there are 128 frequencies generated randomly from the originals 64 frequencies (Figure 176). The signal frequencies are assumed to be uniformly distributed over the bandwidth.

FSK/PSK Target– Parameters	Input Signal	Obtained	Comment
Sampling frequency	15000	15000	Given
Number of phases	5	-	
Number of frequencies	64	-	
Random Hops	128	-	
Bandwidth (Hz)	3000	4500	
SNR	Signal only	-	

Table 42 FSK/PSK Target, phases =128, cycle per phase =5, signal only.

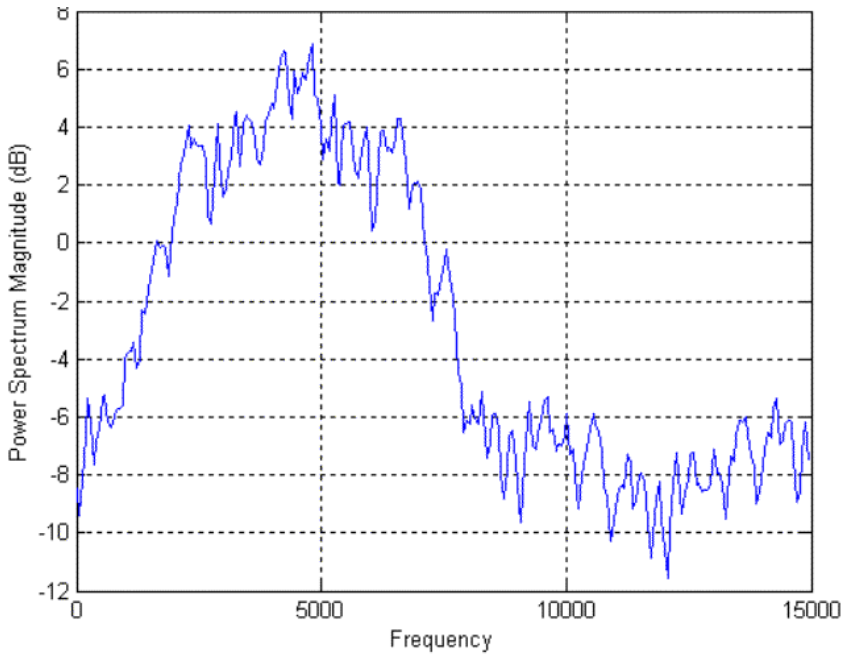


Figure 175 FSK/PSK Target, phases =128, cycle per phase =5, signal only PSD.

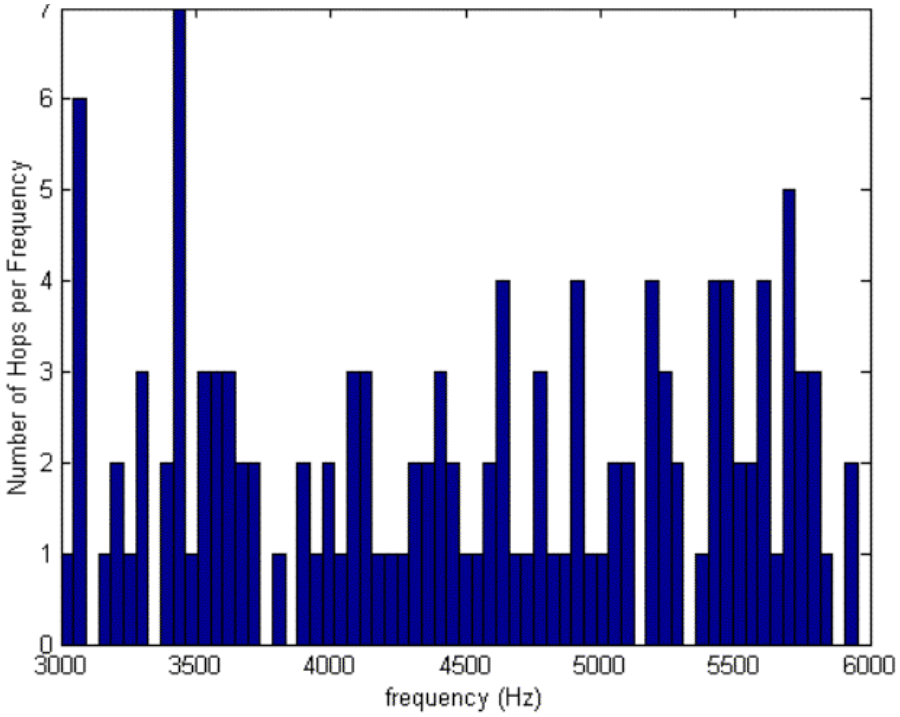


Figure 176 FSK/PSK Target, phases =128, cycle per phase =5, signal only, number of occurrences.

Figure 177 presents the response of the parallel filter arrays, and Figure 178 shows the resulting signal after third-order cumulants estimators are applied to each sub-band frequency. There is not a significant difference between these two figures due to the lack of noise in the input signal. In addition, the power is mostly concentrated between 3 KHz and 6 KHz as expected.

As a result of the particularly efficient LPI characteristics of this modulation, there is not much information that can be extracted from the plots in addition to the bandwidth and the concentration of the power in frequency. The random presentation of frequencies in the signal gives the impression of containing only noise.

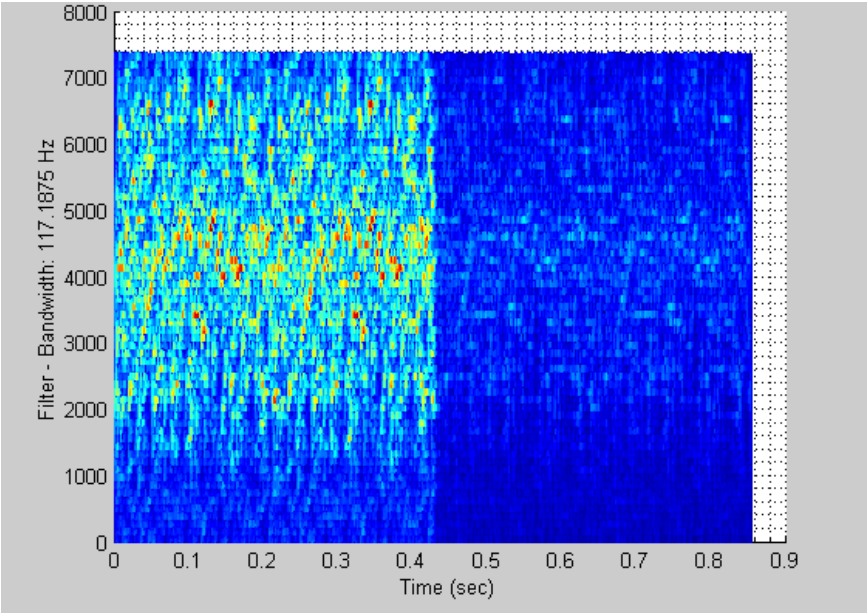


Figure 177 FSK/PSK Target, phases =128, cycle per phase =5, signal only Output of the parallel filter arrays.

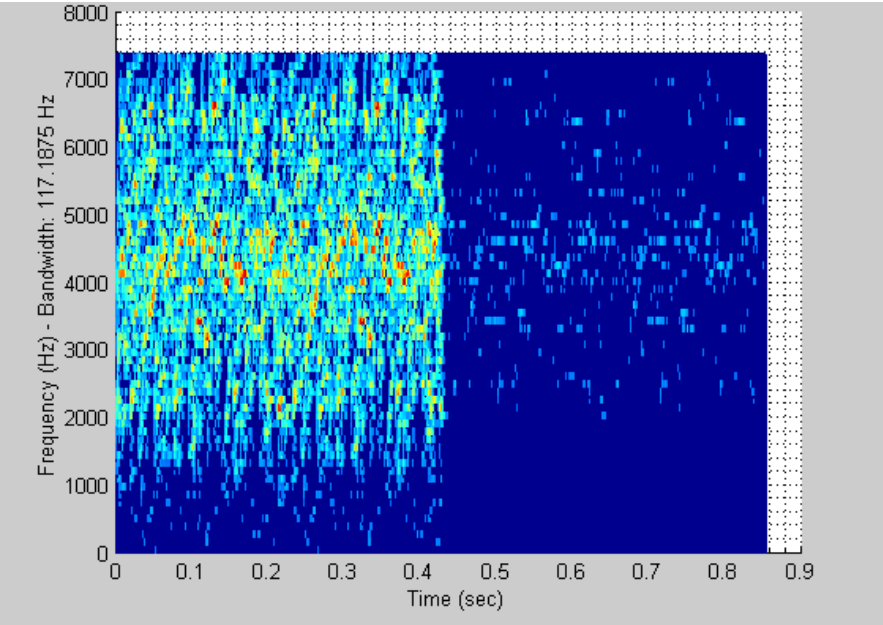


Figure 178 FSK/PSK Target, phases =128, cycle per phase =5, signal only output after HOS.

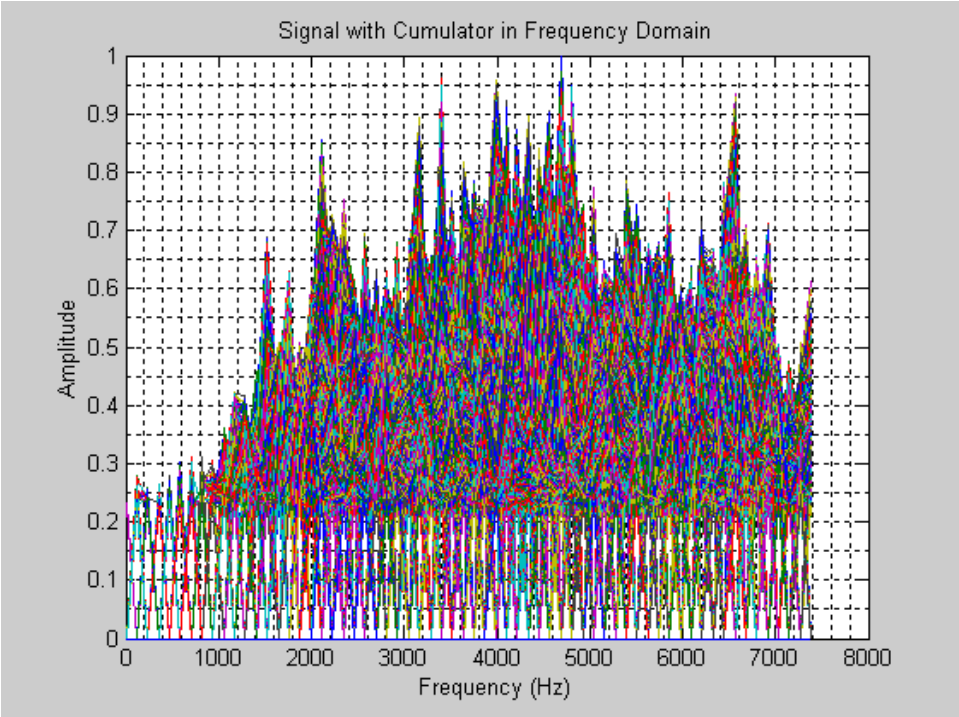


Figure 179 FSK/PSK Target, phases =128, cycle per phase =5, signal only amplitude-frequency plot after HOS.

2. FSK/PSK Target, phases =128, cycle per phase =5, SNR=0 dB

Table 43 describes a FSK/PSK target signal with sampling frequency 15 KHz, 5 different phases, 128 random frequencies and SNR = 0 dB. The PSD of the signal is shown in Figure 180 . This plot presents the distribution of the power along different frequency components of the signal. As a result of the target response there is a concentration of the power between 3 KHZ and 6 KHz.

FSK/PSK Target– Parameters	Input Signal	Obtained	Comment
Sampling frequency	15000	15000	Given
Number of phases	5	-	
Number of frequencies	64	-	
Random Hops	128	-	
Bandwidth (Hz)	3000	3000	
SNR	0	-	

Table 43 FSK/PSK Target, phases =128, cycle per phase =5, SNR=0 dB.

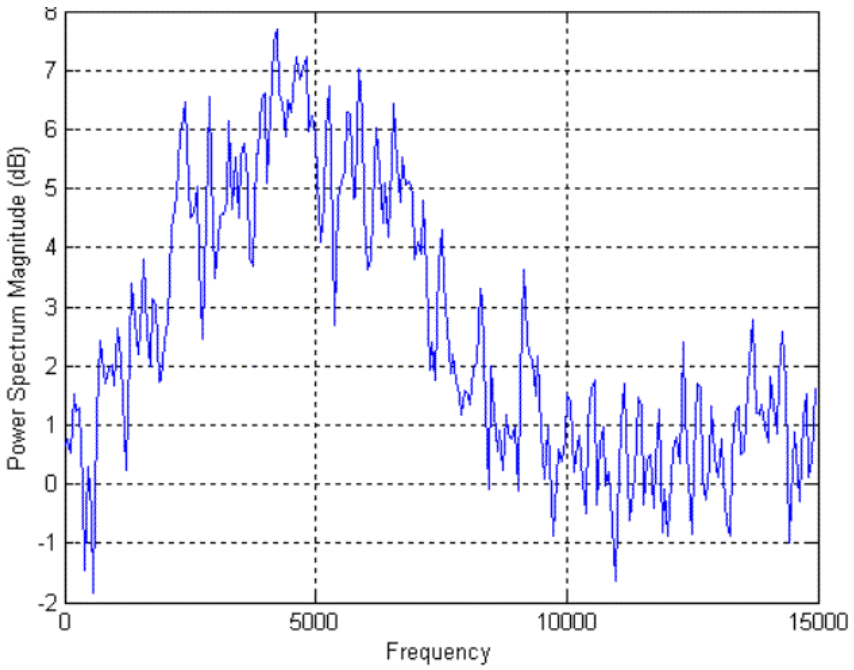


Figure 180 FSK/PSK Target, phases =128, cycle per phase =5, SNR=0 dB.

Figure 181 shows the output of the parallel filter arrays. The input signal is separated into small sub-band signals. Figure 182 presents the resulting signal after HOS. Although most of the white Gaussian noise is suppressed by the third-order cumulant estimators, the signal appears very noisy. The frequencies are completely distributed along the whole bandwidth. This behavior can be confirmed in the Amplitude-frequency plot in Figure 183 , where the complete spectrum is occupied by frequencies.

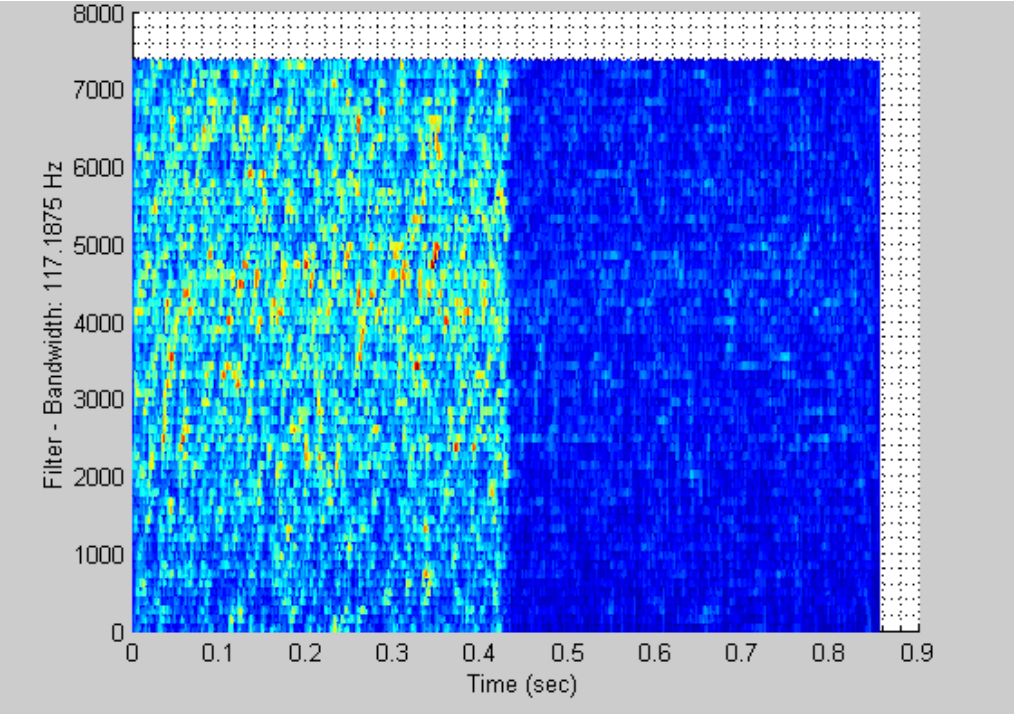


Figure 181 FSK/PSK Target, phases =128, cycle per phase =5, SNR=0 dB output of the parallel filter arrays.

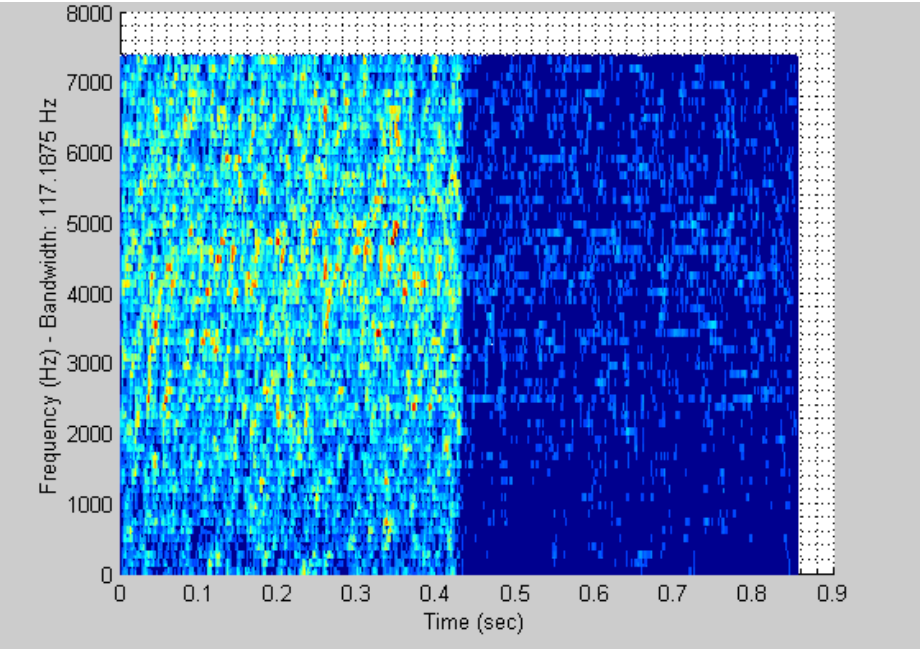


Figure 182 FSK/PSK Target, phases =128, cycle per phase =5, SNR=0 dB output of HOS.

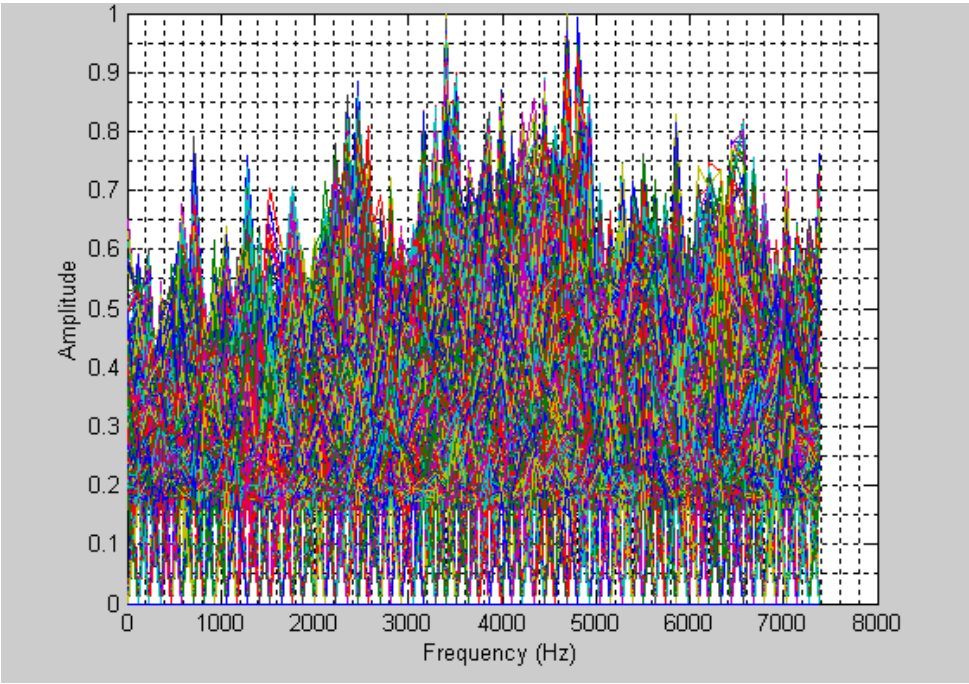


Figure 183 FSK/PSK Target, phases =128, cycle per phase =5, SNR=0 dB amplitude-frequency plot.

3. Summary

Figure 184 shows a summary chart of the performance of the parallel filter arrays and HOS to detect carrier frequency, bandwidth and code period in FSK/PSK target signals analyzed previously. This figure presents a comparison of the parameters in three different environments: signal only (blue), SNR=0 dB (red) and SNR=-6 dB (yellow). The percentage in the chart describes an average of how close is the extracted values from the theoretical values.

After processing the signals, the plots provide imprecise values for carrier frequency, bandwidth and code period in any environment analyzed.

FSK/PSK TARGET	Sequence	Bandwidth (Hz)	Code period (ms)
Signal only	0%	108%	0%
0 dB	0%	88%	0%

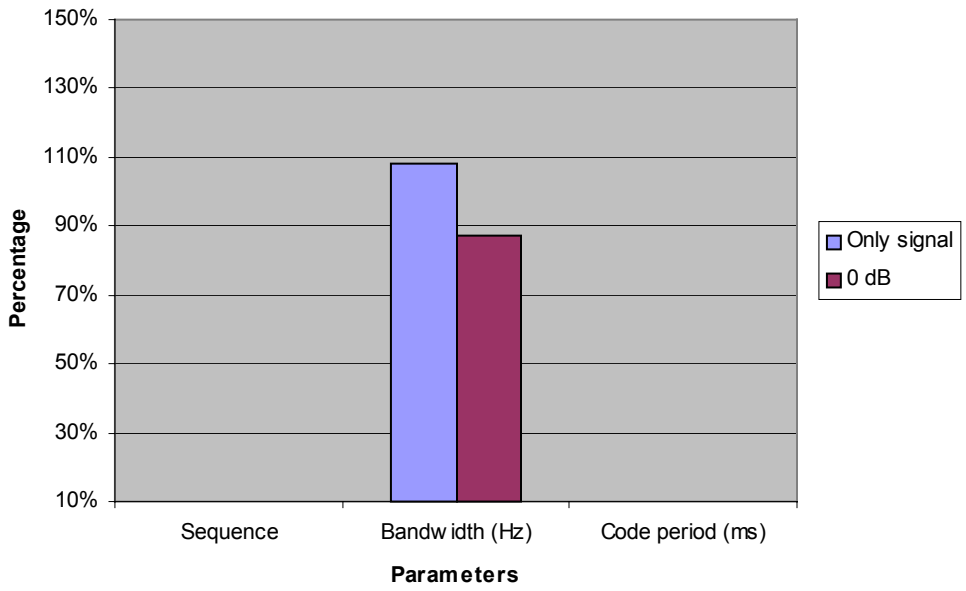


Figure 184 Performance of the signal processing detecting FSK/PSK target signals.

L. COMPARISON OF DIFFERENT POLYPHASE-CODED SIGNALS

The objective of comparing different polyphased-coded signals is to demonstrate the effectiveness of the proposed signal processing to discriminate among different polyphase modulations. One signal per each modulation was selected, having the same carrier frequency (1 KHz)), sampling frequency (7 KHz), number of phases (16) and cycles per phase (5). The results are presented in figures and tables. The most important deviation among the signal is related to the phase shift. The increment between phases produces diverse performance in the modulations, facilitating an accurate identification of each polyphase modulation.

Table 44 shows each one of the analyzed signals and their most important characteristics. In addition, this table presents the resulting phases for the generation of 16 phases (N=4 or N=16). The resulting figures after HOS is provided along with the phase shift plots. The analysis of these plots reveals a close relationship where the conduct of the phases approximate to the frequency-time description of the signal.

Signal	Phase shift values	Comment
Frank	0 0 0 0 0 1.5708 3.1416 4.7124 0 3.1416 6.2832 9.4248 0 4.7124 9.4248 14.1372	Phase changes between adjacent codes are the smallest.
P1	0 -3.1416 6.2832 28.2743 -2.3562 -3.9270 7.0686 30.6305 -4.7124 -4.7124 7.8540 32.9867 -7.0686 -5.4978 8.6394	Lowest code increments from one code element to code element in the center of the waveform
P2	3.5343 1.1781 -1.1781 -3.5343 1.1781 0.3927 -0.3927 -1.1781 -1.1781 -0.3927 0.3927 1.1781 -3.5343 -1.1781 1.1781 3.5343	Symmetric at center frequency
P3	0 0.0123 0.0491 0.1104 0.1963 0.3068 0.4418 0.6013 0.7854 0.9940 1.2272 1.4849 1.7671 2.0739 2.4053 2.7612 (one row)	Largest code element to code element are on middle of the P3 code
P4	0 -2.9452 -5.4978 -7.6576 - 9.4248 -10.7992 -11.7810 -12.3700 -12.5664 -12.3700 -11.7810 -10.7992 -9.4248 -7.6576 -5.4978 -2.9452 (one row)	Largest code element to code element are on the two ends of the P4 code

Table 44 Different Polyphase-coded signals and differences for N=16.

1. Frank Code

The actual Frank-coded signal consists of a carrier (1 KHz), the place of which is modulated according to the indicated baseband waveform sequence. For each frequency or section of the step chirp, a phase group consisting of *N* phase samples is obtained and the total number of code phases is *N*. Note that the phase increments within the four phase groups are 0˚, 90˚, 180˚, and 270˚. However, the phases of the last group are ambiguous (>180˚) and appear as -90˚ phase steps. The last phase group, because of the ambiguity, appears to complete one 360˚. (Figure 185)

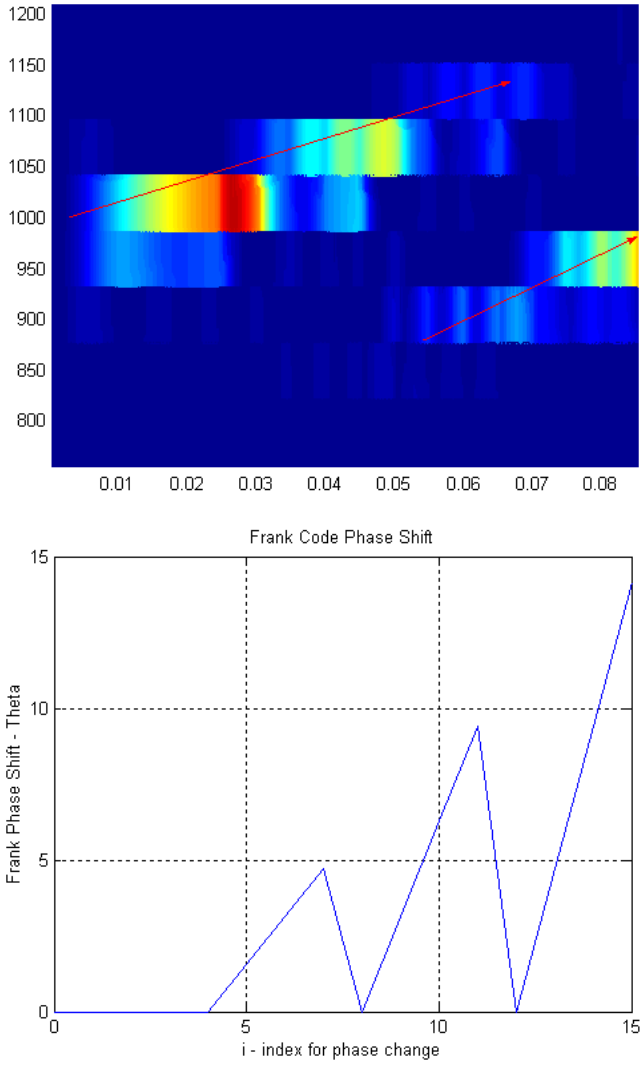


Figure 185 Frank-coded signal: Resulting plot after HOS and phase shift.

2. P1

This code has the lowest code increments from one code element to code element in the center of the waveform as shown in Figure 186 . For a odd number of N, the resulting phases are the same as the Frank code except the phase groups are rearranged. For N even, P1 codes has the same phase increments, within each phase group, as the P2 except the starting phases are different.

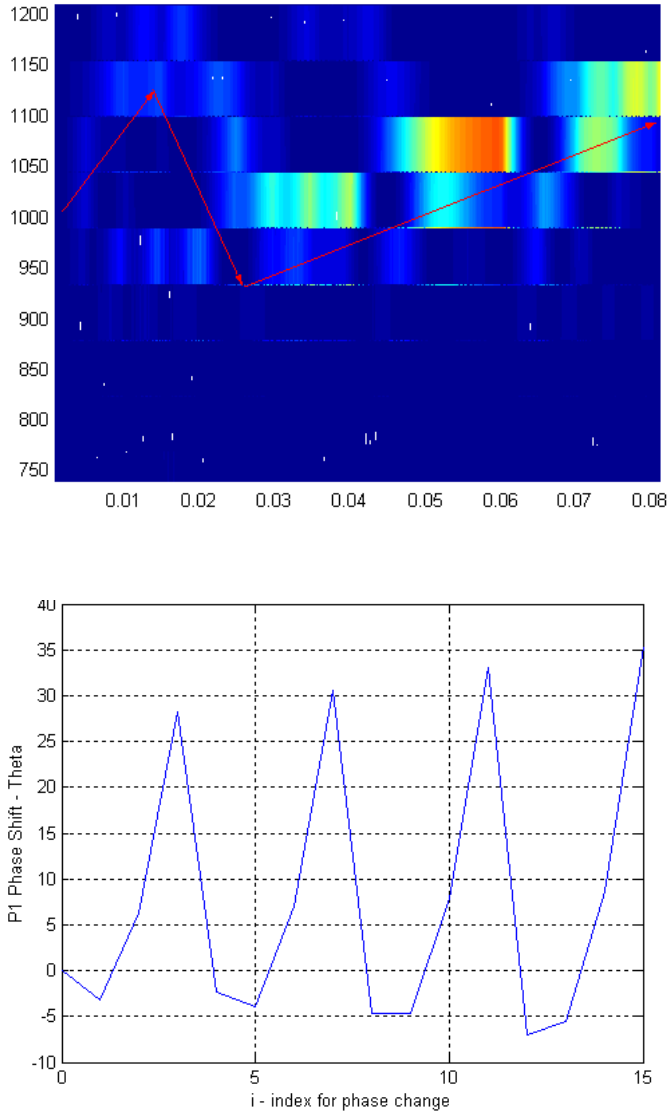


Figure 186 P1-coded signal: Resulting plot after HOS and phase shift.

3. P2

This code is valid for even numbers of phases, and each group of the code is symmetric about zero phase as shown in Figure 187 .

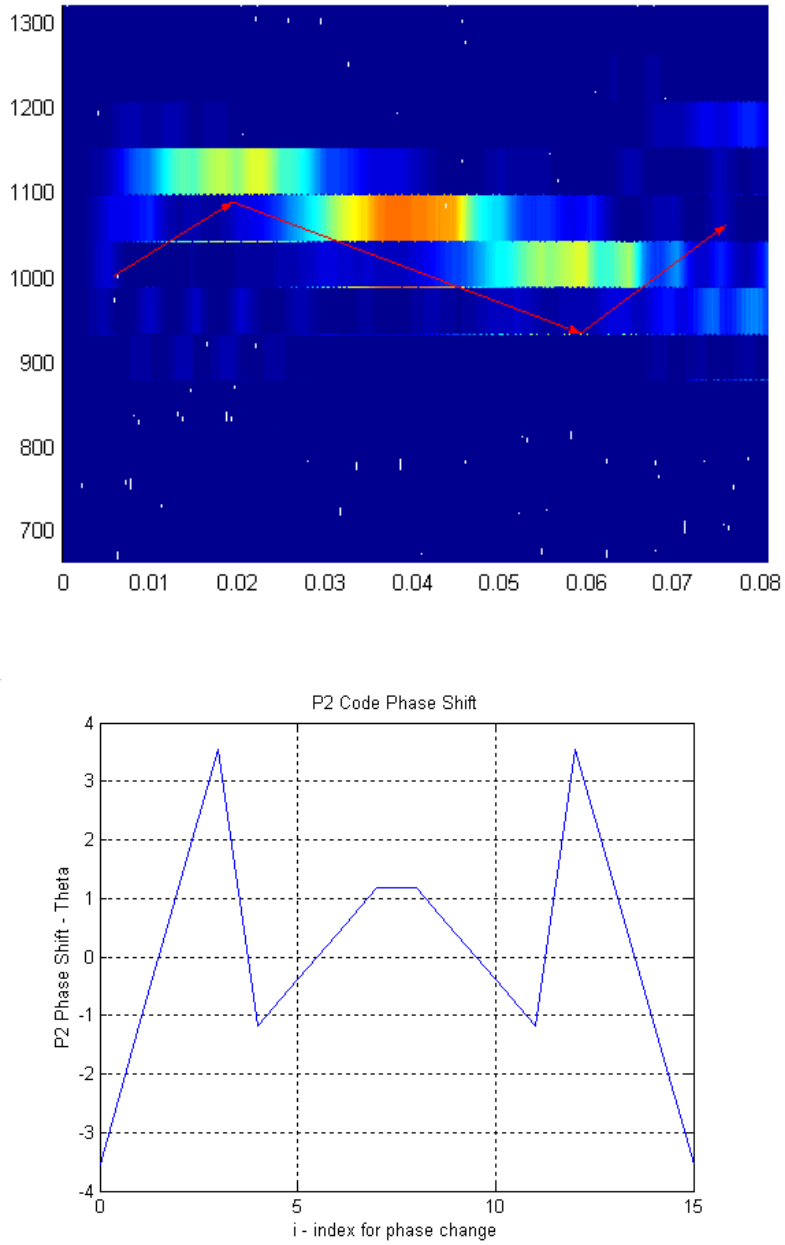


Figure 187 P2-coded signal: Resulting plot after HOS and phase shift.

4. P3

P3 only differs from Frank code by 180° phase shifts every $N^{1/2}$ code elements (one frequency group) and by added phase increments that repeats every $N^{1/2}$ samples (every frequency group). The largest phase increments from code element to code element are in the middle of the P3 code as presented in Figure 188 .

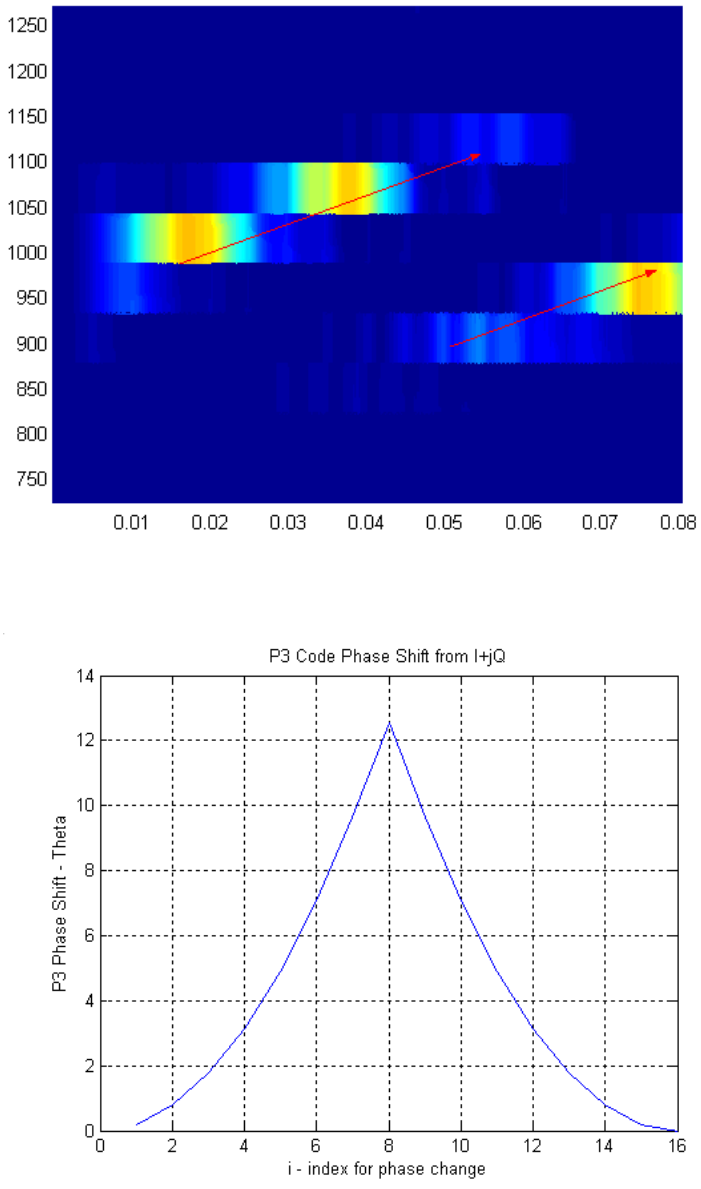


Figure 188 P3-coded signal: Resulting plot after HOS and phase shift.

5. P4

P4 code is very similar to P1 code except that the phase samples are those of a sampled chirped waveform rather than step-chirp waveform. It is noted that the largest phase increment s from code element are on the two ends of the P4 code. Figure 189 describes the resulting signal and its phase shift.

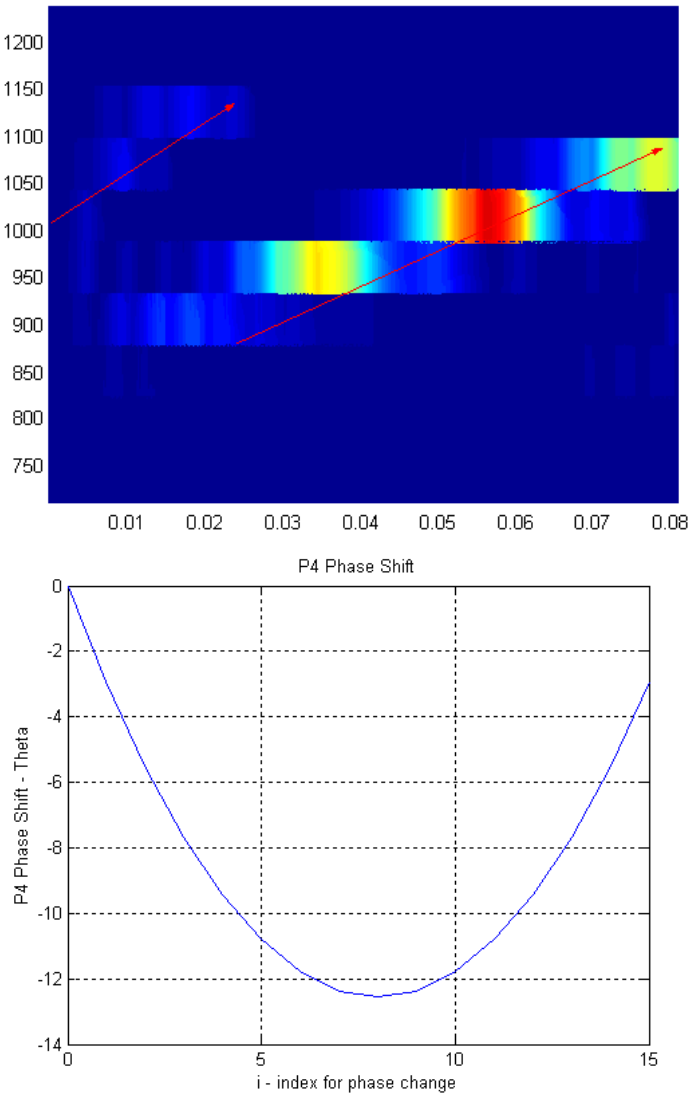


Figure 189 P4-coded signal: Resulting plot after HOS and phase shift.

M. SUMMARY OF EFFECTIVENESS

This section presents a summary of the effectiveness of parallel filter arrays and HOS to detect and extract different parameters from unknown signals for each one of the analyzed modulations. The capability to detect the signal and extract values of parameters is shown in Figure 190 using different colors. Green represents detection with a deviation no greater than ± 10 % from the calculated values; yellow shows detection with a deviation between 10% and 50%, and red represents those measurements whose values exceed 50% from the original values.

This method has demonstrated being highly efficient to detect and extract parameters from unknown LPI signals for FMCW, polyphased-coded signals, Costas signals and FSK/PSK Costas signals. The performance is very low on the detection of FSK/PSK Target where the only parameter detected is the bandwidth.

Modulation	Carrier frequency	Sequence	Bandwidth/Modulation Bandwidth	Code/Modulation period	Code	Phases
BPSK						
FMCW						
FRANK						
P1						
P2						
P3						
P4						
COSTAS						
FSK/PSK/COSTAS						
FSK/PSK TARGET						

Figure 190 Effectiveness on detection for different parameters in each modulation, showing accurate detection (green), poor detection (yellow) and no detection (red).

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

In this document, a proposed signal processing is presented for the detection of LPI radar signals based on the use of a parallel filter arrays and higher order cumulants. This scheme can be related to a time-frequency domain detection method. The detection can be performed without knowing any of the characteristics of the input signal.

The parallel array of filters can approximate the behavior of a matched filter. The purpose of this filter bank is to separate the observed signal into frequency bands. An increment in the number of filters in bank also increases the resolution of the system. The implementation of HOS, particularly third-order cumulant estimators, shows the potential of the method to suppress white Gaussian noise. The detection method also indicates that the third-order cumulant of a signal grows out the third-order cumulant of the noise with increasing SNR.

The efficiency of the proposed method varies with the modulation used in the LPI radar signal. This signal processing demonstrated high efficiency detecting and identifying al the parameters for BPSK and FMCW signals. The detection and identification of Polyphase-coded signal is satisfactory, even though some of the parameters cannot completely be distinguished. This method also exhibits a good discrimination among different polyphase-coded signals as Frank, P1, P2, P3 and P4.

As verified, this technique alone is not sufficient to process the multiplicity of available LPI waveforms, but the combined use of this technique with others, such as Wigner distribution, Cyclostationary processing, and Quadrature mirror filtering will provide the expected response.

B. RECOMMENDATIONS

The MATLAB® code implementing the parallel filter bank and HOS takes a considerable amount of time to produce a result, due to the multiple number of loops in the program. In the future, a more efficient algorithm must be implemented to improve the response time.

One more stage could be created for the automatic recognition of the most important parameters and the modulation of LPI signals. Neural Networks have been used exhaustively in pattern recognition and classification and it seems to be a good approach for feature extraction and an accurate classification of LPI signals.

APPENDIX A. LPI SIGNAL GENERATOR MAIN PROGRAM AND SUB-PROGRAMS

```
%*************************************************************************
% LPIG.m
%
% Use:  This program is the user interface of the LPI signal generator
%
% Inputs:       Type of LPI signal
%
%Output:       In-phase (I) and Quadarture (Q) components of the signal
%              Plots
%
%January 18, 2002
%Maj Fernando Taboada - Venezuelan Army
%*************************************************************************
clc;
clear;
clear all; close all
disp('*********************************************************');
disp('*************PROGRAM TO GENERATE LPI SIGNALS*************');
disp('*********************************************************');
disp(' ')
disp(' ')
disp('')
disp('CHOOSE A TYPE OF CODE TO BE GENERATED:')
disp('1.  BPSK')
disp('2.  FMCW')
disp('3.  FRANK CODE')
disp('4.  POLYPHASE CODE P1')
disp('5.  POLYPHASE CODE P2')
disp('6.  POLYPHASE CODE P3')
disp('7.  POLYPHASE CODE P4')
disp('8.  COSTAS CODE')
disp('9.  FSK/PSK COSTAS')
disp('10. FSK/PSK TARGET')
disp('11. TEST SIGNAL')
disp(' ')
ch=input('Enter a number (1-11): ');
   switch ch
      case 1
         bpsk;
      case 2
         fmcw;
      case 3
         frank;
      case 4
         p1;
      case 5
         p2;
      case 6
         p3;
      case 7
```

```
        p4;
    case 8
        costas;
    case 9
        fsk_psk_costas;
    case 10
        fsk_psk_target;
    case 11
        test_signal;
    end

disp(' ')
disp(' ')
disp('Do you want to generate other LPI signal ? ')
disp(' ')
disp('1. Generate a different LPI')
disp('2. Quit')
again=input('Choose 1 or 2 = ');
switch again
   case 1
      LPIG;
  otherwise
      beep;
      disp('Thanks for using the LPI generator');
   end
```

```
%*******************************************************************************
% bpsk.m
%
% Use:  Code to generated BPSK signals
%
% Inputs:      Amplitud of the carrier signal
%                      Carrier signal frequency - f (Hz)
%            Sampling frequency - fs (HZ)
%            Desired SNR in dB
%            Number of bits per Barker code
%            Number of code periods
%            Number of cycles per Barker bit
%            Number of periods to view on graphs (CW signal)
%
%Output:      In-phase (I) and Quadarture (Q) components of the signal
%            Plots
%
%January 17, 2002
%Maj Fernando Taboada - Venezuelan Army
%*******************************************************************************


clc;clear all;
disp('************************************');
disp('*************** BPSK ***************');
disp('************************************');

%         DEFAULT VARIABLES
A  = 1;                                                 %Amplitud of the carrier signal
f   = 1e3;                                              %Carrier signal frequency - f (Hz)
fs = 7e3;                   %Sampling frequency - fs (Hz)
SNRdb  = 0;                                                     %Desired SNR in dB
barker = 7;                  %Number of bits per Barker code
np   = 175;                                                     %Number of code periods
NPBB = 5;                                                       %Number of cycles per Barker bit
NPV  = 55;                                              %Number of periods to view on graphs (CW signal)

% NEW INPUT
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitude of the carrier signal - (A)= %g.\n', A)
   fprintf('2. Carrier frequency - f (Hz) = %g.\n', f)
   fprintf('3. Sampling frequency - fs (Hz)= %g.\n', fs)
   fprintf('4. Signal to noise ratio - SNRdb (dB)= %g.\n', SNRdb)
   fprintf('5. Number of bits per Barker code - barker (13/11/7)= %g.\n', barker)
   fprintf('6. Number of code periods- np= %g.\n', np)
   fprintf('7. Number of cycles per Barker bit - NPBB= %g.\n',NPBB)
   fprintf('8. Number of code periods to view on graphs= %g.\n', NPV)
   fprintf('9. No changes\n')
   disp(' ')
   option= input('Select a option: ');
```

```
   switch option
   case 1
      A=input('New amplitude of carrier signal (Volts) = ');
   case 2
      f=input('New carrier signal frequency (Hz) = ');
   case 3
      fs=input('New sampling frequency (Hz) = ');
   case 4
      SNRdb=input('New signal to Noise Ratio (dB) = ');
   case 5
      barker=input('New number of bits for Barker Code = ');
   case 6
      np=input('New number of periods of carrier signal to generate = ');
   case 7
      NPBB=input('New number of CW periods per Barker bit = ');
   case 8
      NPV=input('New number of periods to view on graphs = ');
   case 9
      newvar = 0;
   end
    clc;
end

%  Variables below are calculated based upon the above user defined variables
ti = 1/(f*100);                                          %determine suitable time increment to plot carrier signal
t  = 0:ti:np/f;                                            %set up time vector
xt = A*sin(2*pi*f.*t);                %representation of carrier signal(continuous)

SAR=floor(fs/f);
n  = 0:1:SAR*np;                                  %sample vector n, covers np periods of carrier frequency

xnT  = A*cos(2*pi.*n*f/fs);     %create vector of sampled function, for I
xnT2 = A*sin(2*pi.*n*f/fs);   %create shifted version of function, for Q


%************
% This section creates the modulating signal and modulates the carrier
%************
fm = f/NPBB;                                                %set frequency of modulating signal
pw = floor(fs/fm);                                                %determine # of samples (of n) per
Barker bit

%create sequence of the (7, 11 or 13)-bit Barker code with pw samples of each bit,
%string several sequences end to end to match length of sampled signal

if barker==13
   brk = [ones(1,pw*5),-(ones(1,pw*2)),ones(1,pw*2),-ones(1,pw),ones(1,pw),-ones(1,pw),ones(1,pw)];%13 bit
elseif barker==11
   brk = [ones(1,pw*3),-(ones(1,pw*3)),ones(1,pw),-ones(1,pw),-ones(1,pw),ones(1,pw),-ones(1,pw)];% 11 bit
else
   brk = [ones(1,pw*3),-(ones(1,pw*2)),ones(1,pw),-ones(1,pw)];% 7 bits
end

brkseq = [];                                                %initialize brkseq to hold a sequence of brk
ns = floor(length(n)/(pw*barker)); %integer number of complete 13-bit Barker sequences
```

```
for step = 1:1:ns;                                  %create multiple n-bit Barker sequence vector
  brkseq = [brkseq,brk];
end

%************
% Here is where the modulation is done and the I and Q channels created
%************

I  = xnT(1:barker*ns*pw).*brkseq;     %modulate sampled carrier with Barker sequence, I
Q  = xnT2(1:barker*ns*pw).*brkseq;          %modulate shifted function with Barker sequence, Q
rl  = length(I);                                                    %get length of the new vector

%******************
% In this section we create White Gaussian Noise (WGN) and add it to both the
% I and the Q channels.  In the original program the noise was developed using
% the normrnd MATLAB function. To standarize noise for all the programs, we
%use the wgn MATLAB function.
%******************

[a,b]=size(xnT);
SNR=10^(SNRdb/10);
power=10*log10(A^2./(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power); %calculate noise at specified SNR
IwN =I+noise(1:length(I));          %add noise to I
QwN=Q+noise(1:length(Q));          %add noise to Q
 xnTwN = xnT + noise;                                          %add WGN to sampled signal, for reference
only

%Generation of plots
disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
   disp(' ')
     %*************************
      % Plot a CW representation of the signal and a sampled version of the signal
      %*************************
      figure(1);
      %plot carrier frequency (CW) and sampled function
      subplot(2,1,1);                                  %CW representation

      if ( np > NPV );                                  %just show some of the data to keep screen uncluttered
         plot(t(1:1:100*NPV),xt(1:1:100*NPV));grid;
         axis([0 0.012 -A*1.5 A*1.5]);
         title(['Continuous Wave Signal, f = ',num2str(f),', #periods= ',num2str(np),', ',num2str(NPV),' shown']);
      else
         plot(t,xt);  grid;                          %plot carrier frequency as continuous function

         title(['Continuous Wave Signal, f = ',num2str(f),', #periods= ',num2str(np)]);
      end

      xlabel('Time (sec)'),ylabel('Amplitude');


      subplot(2,1,2);                                  %show sampled version of carrier signal
```

```
    if ( np > NPV )                                    %keep display uncluttered
       stem(n(1:1:SAR*NPV),xnT(1:1:SAR*NPV));grid;%plot sampled carrier frequency (portion of)
       axis([1 SAR*NPV -A*1.5 A*1.5]);
       title(['Sampled signal, fs = ',num2str(fs),', #periods= ',num2str(np),', ',num2str(NPV),' shown']);
    else
       stem (n,xnT);grid;                              %plot sampled carrier frequency
       title(['Sampled signal, fs = ',num2str(fs),', #periods= ',num2str(np)]);
    end

    xlabel('n (sample #)'),ylabel('x[n]');
    title(['Sampled signal, fs = ',num2str(fs),', #periods= ',num2str(np)]);

    %***********************
    %Plot Modulating signal and PSD
    %***********************
    figure(2);
    subplot(2,1,1);                                    %show modulating signal, distinguishing
between bits
    set (gca,'YLim',[1.2*min(brk) 1.2*max(brk)]);%force y axis limits to better see signal
    xlabel('n (sample #)'),ylabel('Amplitude');
    if ( np > NPV )                                    %limit displayed amount
       flag = 1;                                       %flag tracks bit changes to change display

       for step=1:pw:SAR*NPV                           %step through each Barker bit
          if ( flag == 1 )                         %change plotting for each bit
             hold on,plot(n(step:1:step+pw-1),brkseq(step:1:step+pw-1));grid;
          else
             hold on,plot(n(step:1:step+pw-1),brkseq(step:1:step+pw-1),'x');grid
          end

       flag = -flag;                                   %swap flag
       end
       step=ceil(step/pw);                        %determine # of bits displayed
       title([num2str(barker),'-bit Barker Sequence',' # periods shown = ' num2str(step/barker)]);
    else
       hold on, plot(n(1:rl),brkseq(1:rl),'x');grid;%plot the modulating signal
       title([num2str(barker),'-bit Barker Sequence']);
    end
    subplot(2,1,2);          %show Power Spectral Density of the modulating signal
    psd(brkseq,[],fs);
    title(['Power Spectral Density of the', num2str(barker),'-bit Barker Sequence', ' # samples =
',num2str(length(brkseq))]);

    %***********************
    %Plot the sampled signal and the Barker sequence in subplot 1, then show the
    % results after modulation in subplot 2
    %***********************
    figure(3);
    subplot(2,1,1);

    if ( np > NPV )                                    %show only as much as the user asked for
       stem(n(1:1:SAR*NPV),xnT(1:1:SAR*NPV));
       axis([0 SAR*NPV -A*1.5 A*1.5]);
       hold on, plot(n(1:1:SAR*NPV),brkseq(1:1:SAR*NPV),'r--');%modulating signal in red
    else
```

```
      stem(n(1:rl),xnT(1:rl));                                    %plot the sampled carrier
signal
      axis([0 SAR*NPV -A*1.5 A*1.5]);
      hold on, plot(n(1:rl),brkseq(1:rl),'r--');%plot the modulating signal in red
   end

   xlabel('n (sample #)'),ylabel('Amplitude');
   title([num2str(barker),'-bit Barker Sequence Overlayed on Sampled Signal',', NPBB= ',num2str(NPBB)]);

   subplot(2,1,2);                                             %show results after modulation (I signal)

   if ( np > NPV )
      stem(n(1:1:SAR*NPV),I(1:1:SAR*NPV));grid
      axis([0 SAR*NPV -A*1.5 A*1.5]);
   else
      stem(n(1:rl),I);grid;                                    %plot the modulated signal
      axis([0 SAR*NPV -A*1.5 A*1.5]);
   end
   xlabel('n (sample #)'),ylabel('Amplitude');
   title(['Sampled Signal Modulated by',num2str(barker),'-bit Barker Sequence',' NPBB= ',num2str(NPBB)]);

   %**********************
   %PSD of the carrier frequency before and afeter modulation
   %**********************
   figure(4);
   subplot(2,1,1);                                             %plot PSD of unmodulated carrier
   psd(xnT,[],fs);
   xlabel('Frequency (Hz)');
   title(['PSD of Original Sampled Function, fs = ',num2str(fs),', # samples= ',num2str(rl)]);


   subplot(2,1,2);                                             %plot PSD of modulated carrier (I)
   psd(I,[],fs);
   xlabel('Frequency (Hz)');
   title(['PSD of Modulated Function, fs = ',num2str(fs),', # samples= ',num2str(rl),', NPBB=
',num2str(NPBB)]);

   %**********************
   %Show statistics and portion of the noise vector
   %**********************
   figure(5);
   subplot(2,1,1);                                                                %show noise
signal vs time (sample)

   if (length(noise) > 100 );
   plot(noise(1:100));grid;
   title(['Plot of Noise Signal, channel I, # samples= ',num2str(rl),', only 100 shown']);
   else
   plot(noise);grid;                                           %look at complete noise
vector
   title(['Plot of Noise Signal, channel I, # samples= ',num2str(rl)]);
   end

   xlabel('Sample #'),ylabel('Magnitude');
```

```
    subplot(2,1,2);                                          %show histogram, mu, and sigma of
noise in I
    hist(noise,30);                                          %show that the noise has normal
distribution
    xlabel('Magnitude'),ylabel('Frequency');
    title(['Histogram of Noise Signal, channel I']);

    %**********************
    %Compare I signal without noise to I signal with noise
    %**********************
    % Figure 6 shows how the noise affects the signal.  Depending upon parameter
    % settings, the user may wish to use the plot commands instead of the stem
    % commands that are used.
    figure(6);
    subplot(2,1,1);                                          %show I signal without noise

    if ( np > NPV )
       stem(n(1:1:SAR*NPV),I(1:1:SAR*NPV));grid;
       axis([0 SAR*NPV -A*1.5 A*1.5]);
       %plot(n(1:1:SAR*NPV),I(1:1:SAR*NPV),'x');
    else
       stem(n(1:rl),I);grid;                          %plot the modulated signal
       axis([0 SAR*NPV -A*1.5 A*1.5]);
    end

    xlabel('n (sample #)'),ylabel('Amplitude');
    title(['Sampled Signal Modulated by', num2str(barker),'-bit Barker Sequence',', NPBB=
',num2str(NPBB)]);


    subplot(2,1,2);                                          %show I signal after noise has been
added

    if ( np > NPV )
       stem(n(1:1:SAR*NPV),IwN(1:1:SAR*NPV));grid;
       %plot(n(1:1:SAR*NPV),IwN(1:1:SAR*NPV),'x');
    else
       stem(n(1:rl),IwN); grid;                              %plot the modulated signal with noise
added
    end

    xlabel('n (sample #)'),ylabel('Amplitude');
    title(['Sampled Signal Modulated by', num2str(barker),'-bit Barker Sequence, WGN added',', SNR=
',num2str(SNRdb ,3),'dB']);

    %******************
    % Now, simply to gain a better understanding of how the same noise impacts the PSD
    % of the modulated signal and the unmodulated signal, show a PSD of the unmodulated
    % signal once noise has been added to it and a PSD of the modulated signal with the
    % same noise vector added.

    figure(7);
    subplot(2,1,1);                                          %PSD of unmodulated signal with noise added
    psd(xnTwN,[],fs);
    title(['PSD of Unmodulated Signal with Noise, fs = ',num2str(fs),', # samples= ',num2str(rl),', NPBB=
',num2str(NPBB),', SNR= ',num2str(SNRdb,2),'dB']);
```

```
subplot(2,1,2);                                          %PSD of modulated signal with noise added
psd(IwN,[],fs);
title(['PSD of Modulated Function with Noise, fs = ',num2str(fs),', # samples= ',num2str(rl),', NPBB=
',num2str(NPBB)]);


%******************
% In this section we create two types of matched filters and analyze the results
% of each after running the modulated signal (without noise) through them.  One filter
% is designed as the standard integrate and dump (reverse the original filter, run the
% modulated signal through it, multiplying the overlapping bits of the two sequences
% together and summing them up for each step through the filter).  This filter is
% primarily used to find a binary signal that has been encoded in the carrier, but
% it doesn't work well when trying to find the carrier signal.  The second filter is
% similar to the first, but once the multiplication of overlapping bits is completed,
% the individual products are then multiplied by the bits of a clean sampled version
% of the carrier signal.  The individual products are then added together and stored
% as the comparison value for that step (sample #) in the filter sequence.
%******************

% 1st, duplicate the modulated signal and pad it with leading and trailing zeros
xmm = [zeros(1,length(brk)-1) I zeros(1,length(brk)-1)];
mf  = [brk(length(brk):-1:1)];%reverse n-bit Barker code to create matched filter

% 2nd, get sequence of original sample and reverse it for tailored filter
tf = xnT(1:length(brk));                    %get 1 complete sequence of original signal for comparison
tf = [tf(length(tf):-1:1)];%reverse the order to match with the filtered signal

% initialize vectors to store the result
cmtdump = [zeros(1,ns*length(brk)/pw)];%vector to store traditional filter results
cmt = [zeros(1,ns*length(brk)/pw)];              %vector to store tailored filter results

for step = 1:1:ns*length(brk);                   %step through the sequence
   temp = xmm(step:step+length(brk)-1);%get n Barker bits worth of samples
          temp = [temp(length(temp):-1:1)];      %reverse order to run through matched filter
   temp = temp.*mf;                                                    %put against matched filter
   cmtdump( step ) = sum(temp);                          %integrate and dump method
   temp = temp.*tf;                                                    %compare with tailored filter
   cmt( step ) = abs(sum(temp));                         %store results of tailored filter
end

%normalize the results so that the maximum value = 1
cmtdump = cmtdump/max(cmtdump);
cmt = cmt/max(cmt);
figure(8);                          %show the results of each of the filters

subplot(2,1,1);          %show results of traditional filter
if ( step-1 > 3*length(brk)/pw)
   plot(10*NPBB:3*length(brk),cmtdump(10*NPBB:3*length(brk)));grid;
else
   plot(1:step-1,cmtdump);grid;
end
xlabel('Sample #'),ylabel('Strength (normalized)');
title(['Comparison with Original after Traditional Matched Filter, no noise added']);
```

```
%The next line of code sets the x-axis tick marks to occur where the samples should
%have the closest match to the original signal.  This will be the product of the
%length of the Barker code (n), the sample rate (SAR), and the number of periods of
% of the signal per Barker bit (NPBB), which is equal to the length of brk.
set (gca,'XTick',[length(brk):length(brk):ns*length(brk)]);


subplot(2,1,2);          %show results of tailored filter
if (step-1 > 3*length(brk)/pw)
   plot(10*NPBB:3*length(brk),cmt(10*NPBB:3*length(brk)));grid;
else
   plot(1:step-1,cmt);grid
end
xlabel('Sample #'),ylabel('Strength (normalized)');
title(['Comparison with Original after Tailored Matched Filter, no noise added']);
set (gca,'XTick',[length(brk):length(brk):ns*length(brk)]);


%***************
% Apply both filters like above, but this time to the signal with noise added
%***************
xmmn = [zeros(1,length(brk)-1) IwN zeros(1,length(brk)-1)];        %duplicate and pad modulated vector
cmt  = [zeros(1,ns*length(brk)/pw)];
cmtdump = [zeros(1,ns*length(brk)/pw)];

for step = 1:1:ns*length(brk);                        %step through each Barker bit (pw sample separation)
   temp = xmmn(step:step+length(brk)-1);%get n bits worth of samples
   temp = [temp(length(temp):-1:1)];     %reverse the order to run through matched filter
   temp = temp.*mf;                                                          %put against matched filter
   cmtdump( step ) = sum(temp);                                %traditional integrate and dump filter results
   temp = temp.*tf;                                                          %compare with tailored filter
   cmt( step ) = abs(sum(temp));                               %accumulate tailored filter results
end

%normalize the results so that maximum = 1
cmtdump = cmtdump/max(cmtdump);
cmt = cmt/max(cmt);
figure(9);                          %show results of each of the filters
subplot(2,1,1);          %show results of traditional filter
if ( step-1 > 2.2*SAR*NPBB*length(brk)/(pw))

plot(10*NPBB:floor(2.2*SAR*NPBB*length(brk)/(pw)),cmtdump(10*NPBB:floor(2.2*SAR*NPBB*length(br
k)/(pw))));grid;
   else
      plot(1:step-1,cmtdump);grid;
   end

   xlabel('Sample #'),ylabel('Strength (normalized)');
   title(['Comparison with Original after Traditional Matched Filter, with noise, SNR=
',num2str(SNRdb,2),'dB']);
   set (gca,'XTick',[length(brk):length(brk):ns*length(brk)]);


   subplot(2,1,2);          %show results of tailored filter
   if (step-1 > 2.2*SAR*NPBB*length(brk)/(pw))
```

```
plot(10*NPBB:floor(2.2*SAR*NPBB*length(brk)/(pw)),cmt(10*NPBB:floor(2.2*SAR*NPBB*length(brk)/(pw))));
          grid;
       else
          plot(1:step-1,cmt);grid
       end

       xlabel('Sample #'),ylabel('Strength (normalized)');
       title(['Comparison with Original after Tailored Matched Filter, with noise, SNR= ',num2str(SNRdb,2),'dB']);
       set (gca,'XTick',[length(brk):length(brk):ns*length(brk)]);
else
   disp('Signal not plotted')
   fprintf('\n\n')
end
% In order to get a standard format of the data coming from this program and the P4, FMCW the I and Q channels
% are saved as I and Q with the Noise included
format short e;
IwN=IwN';
QwN=QwN';
disp(' ')

plt2 = input('Do you want to save the new signal (Y/y or N/n) ?','s');
disp(' ')
if (plt2 == 'Y') | (plt2 =='y')

   I2=I; Q2=Q;
   I=IwN; Q=QwN;
   ff=floor(f/1e3);
   ffs=floor(fs/1e3);
   save(['B_', num2str(ff), '_', num2str(ffs), '_', num2str(barker), '_', num2str(NPBB),'_',num2str(SNRdb)],'I','Q');
   I=I2';
   Q=Q2';
   save(['B_' num2str(ff) '_' num2str(ffs) '_' num2str(barker) '_' num2str(NPBB) '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  B_', num2str(ff), '_', num2str(ffs), '_', num2str(barker), '_', num2str(NPBB),'_',num2str(SNRdb)]);
   disp(['Signal only save as :  B_', num2str(ff), '_', num2str(ffs), '_', num2str(barker), '_', num2str(NPBB),'_s']);
   disp(['Directory:  ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

```
%*************************************************************************
% fmcw.m
%
% Use:  Code to generated triangular FMCW signals
%
% Inputs:      Amplitud of the carrier signal
%                       Carrier signal frequency - f (Hz)
%             Sampling frequency - fs (HZ)
%             Desired SNR in dB
%             Modulation bandwidth
%             Modulation period
%             Number of triangles in the signal
%
%
%Output:       In-phase (I) and Quadarture (Q) components of the signal
%             Plots
%
%January 17, 2002
%Maj Fernando Taboada - Venezuelan Army
%*************************************************************************

clc;clear all;
disp('*********************************************');
disp('****************** FMCW ********************');
disp('*********************************************');

%          DEFAULT VARIABLES
A  = 1;                                                    %Amplitud of the carrier signal
f0 =1e3;                        %Carrier signal frequency -  (Hz)
fs = 7e3;                               %Sampling frequency
SNR_dB  = 0;                                               %Desired SNR in dB
deltaF = 250;                                          %Modulation bandwidth
tm = 0.05;                                                 %Modulation period
triangles=5;                     %No of triangles to be generated
sigma=1;

% NEW INPUT
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitud of the carrier signal - A= %g.\n', A)
   fprintf('2. Carrier frequency - f0 (Hz)= %g.\n', f0)
   fprintf('3. Sampling frequency - fs (Hz)=%g.\n',fs)
   fprintf('4. Signal to noise ratio - SNR_dB (dB)= %g.\n', SNR_dB)
   fprintf('5. Modulation bandwidth - deltaF (Hz)= %g.\n', deltaF)
   fprintf('6. Modulation period - tm (sec)= %g.\n',tm)
   fprintf('7. Number of triangles - triangles=%g.\n',triangles)
   fprintf('8. No changes\n')
   disp(' ')
   option= input('Select a option: ');
   switch option
   case 1
      A=input('New Amplitud of the carrier signal= ');
   case 2
```

```
      f0=input('Carrier frequency (Hz)=');
   case 3
      fs=input('New Sampling frequency (Hz)= ');
   case 4
      SNR_dB=input('New Signal to noise ratio (dB)= ');
   case 5
      deltaF=input('New modulation bandwidth (Hz)= ');
   case 6
      tm=input('New modulation period (sec)= ');
   case 7
      triangles =input('How many triangles (4/5/6) = ');
   case 8
      newvar = 0;
   end
   clc;
end


if SNR_dB ~= -inf
  SNR = 10^(SNR_dB/20);                    %20log since we're dealing with voltage
  sigma = A/(sqrt(2)*SNR);                 %Std. Dev. of noise required to achieve input SNR
else
  SNR = 0;                                                    %For noise only, SNR = 0
  sigma = 1;                                                  %If user wants noise only, let Std Dev = 1
end

ts = 1/fs;                                 %Sample Period
time = (0:ts:tm)';  %Vector of sample times

%Build the pure sin & cos waveforms
I_carrier = cos(2*pi*f0.*time); % Carrier signals
Q_carrier = sin(2*pi*f0.*time); % Carrier signals



f1 = f0 - deltaF/2 + deltaF/tm.*time;      %Up-ramp side of triangle
f2 = f0 + deltaF/2 - deltaF/tm.*time;      %Down-ramp side of triangle

%To represent different numbers of triangles
if triangles==1
   f = 1e-6*[f1;f2];
   elapsed_time = 1e6.*(linspace(0,2*tm,length(f)))';
elseif triangles ==4
   f = 1e-6*[f1;f2;f1;f2;f1;f2;f1;f2];
   elapsed_time = 1e6.*(linspace(0,8*tm,length(f)))';  %time vector used to plot f

                                           %time is in microseconds
elseif triangles ==5
   f = 1e-6*[f1;f2;f1;f2;f1;f2;f1;f2;f1;f2];          %produces train of 5 triangle pulses
   elapsed_time = 1e6.*(linspace(0,10*tm,length(f)))';
else triangles ==6                         %frequencies are in MHz
   f = 1e-6*[f1;f2;f1;f2;f1;f2;f1;f2;f1;f2;f1;f2];
   elapsed_time = 1e6.*(linspace(0,12*tm,length(f)))';
end
```

```
%sI1 is the In-Phase (I) transmitted signal for the up-ramp (without noise)
%sI2 is the In-Phase (I) transmitted signal for the down-ramp (without noise)
if SNR ~= 0
  sI1 = A*cos(2*pi*((f0-deltaF/2).*time + deltaF/(2*tm).*time.^2));
  sI2 = A*cos(2*pi*((f0+deltaF/2).*time - deltaF/(2*tm).*time.^2));
else
  sI1 = zeros(length(time),1);                    %If noise only, Signal = 0
  sI2 = zeros(length(time),1);
end

%Next two lines add req'd noise level to up & down ramps (I channel).
%Used to plot time sequence & PSD for each ramp
sI1_noisy = sI1 + sigma*randn(length(sI1),1);
sI2_noisy = sI2 + sigma*randn(length(sI2),1);


%Creates the I-channel time sequence of four up&down ramp pairs (NO noise)
I = [sI1;sI2;sI1;sI2;sI1;sI2;sI1;sI2];

%Adds the required noise to the time sequence

[a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power);%calculate noise at specified SNR
IwN=I+noise;              %add noise to I with P4 phase shift


%sQ1 is the Quadrature (Q) transmitted signal for the up-ramp (without noise)
%sQ2 is the Quadrature (Q) transmitted signal for the down-ramp (without noise)
if SNR ~= 0
  sQ1 = A*sin(2*pi*((f0-deltaF/2).*time + deltaF/(2*tm).*time.^2));
  sQ2 = A*sin(2*pi*((f0+deltaF/2).*time - deltaF/(2*tm).*time.^2));
else
  sQ1 = zeros(length(time),1);                    %If noise only, Signal = 0
  sQ2 = zeros(length(time),1);
end

%Next two lines add req'd noise level to up & down ramps (Q channel).
%Used to plot time sequence & PSD for each ramp
sQ1_noisy = sQ1 + sigma*randn(length(sQ1),1);
sQ2_noisy = sQ2 + sigma*randn(length(sQ2),1);



%Creates the Q-channel time sequence of four up&down ramp pairs (NO noise)
Q = [sQ1;sQ2;sQ1;sQ2;sQ1;sQ2;sQ1;sQ2];

%Adds the required noise to the time sequence

QwN=Q+noise;

%******************************************************
%PLOTS
%******************************************************
```

```
disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
   disp(' ')
   %Plots time sequence & PSDs of In-Phase (cos) waveform
   figure;
   subplot(2,1,1);
   plot(time(1:50),I_carrier(1:50)); grid;
   title('Pure In-Phase Sinusoidal Carrier');
   xlabel('Time (s)'); ylabel('Cos(2*pi*f_0*t)');
   subplot(2,1,2);
   psd(I_carrier,[],fs);
   title('Power Spectral Density of Pure In-Phase Carrier');

   %Plots time sequence & PSDs of Quadrature (sin) waveform
   figure;
   subplot(2,1,1);
   plot(time(1:50),Q_carrier(1:50)); grid;
   title('Pure Quadrature Sinusoidal Carrier');
   xlabel('Time (s)'); ylabel('-Sin(2*pi*f_0*t)');
   subplot(2,1,2);
   psd(Q_carrier,[],fs);
   title('Power Spectral Density of Pure Quadrature Carrier');

   %Plots the triangular modulating waveform
   figure;
   plot(elapsed_time,f); grid;
   title('Triangular Modulating Signal')
   xlabel('Time (us)'); ylabel('Frequency (MHz)');

   %Plots time sequence & PSD for I-channel up ramp
   figure;
   subplot(2,1,1);
   plot(time, sI1_noisy); grid;
   title(['In-Phase, Up-Ramp Transmitted Signal - SNR = ', num2str(SNR_dB), ' dB'])
   xlabel('Time (s)'); ylabel('Signal');
   subplot(2,1,2);
   psd(sI1_noisy,[],fs);
   title('Power Spectral Density of In-Phase, Up-Ramp Segment');


    %Plots time sequence & PSD for I-channel down ramp
   figure;
   subplot(2,1,1);
   plot(time, sI2_noisy); grid
   title(['In-Phase, Down-Ramp Transmitted Signal- SNR = ', num2str(SNR_dB), ' dB'])
   xlabel('Time (s)'); ylabel('Signal');
   subplot(2,1,2);
   psd(sI2_noisy,[],fs);
   title('Power Spectral Density of In-Phase, Down-Ramp Segment');


   %Plots time sequence & PSD for Q-channel up ramp
   figure;
   subplot(2,1,1);
```

```
   plot(time, sQ1_noisy);grid;
   title(['Quadrature, Up-Ramp Transmitted Signal - SNR = ', num2str(SNR_dB), ' dB'])
   xlabel('Time (s)'); ylabel('Signal');
   title(['Quadrature, Up-Ramp Transmitted Signal (Near End) - SNR = ', num2str(SNR_dB), 'dB'])
   xlabel('Time (s)'); ylabel('Signal');
   subplot(2,1,2);
   psd(sQ1_noisy,[],fs);
   title('Power Spectral Density of Quadrature, Up-Ramp Segment');


   %Plots time sequence & PSD for Q-channel down ramp
   figure;
   subplot(2,1,1);
   plot(time, sQ2_noisy); grid;
   title(['Quadrature, Down-Ramp Transmitted Signal - SNR = ', num2str(SNR_dB), 'dB'])
   xlabel('Time (s)'); ylabel('Signal');
   subplot(2,1,2);
   psd(sQ2_noisy,[],fs);
   title('Power Spectral Density of Quadrature, Down-Ramp Segment');

else
   disp('Signal not plotted')
   fprintf('\n\n')
end


%Saves the I & Q vectors in a .mat file for processing by the receiver folks
disp(' ')
saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');
if (saveresult == 'Y') | (saveresult =='y')
   I2=I; Q2=Q;
   I=IwN; Q=QwN;
   ff=floor(f0/1e3);
   ffs=floor(fs/1e3);
   save(['F_', num2str(ff), '_', num2str(ffs), '_', num2str(deltaF), '_', num2str(tm/1000),
'_',num2str(SNR_dB)],'I','Q');
   I=I2;
   Q=Q2;
   save(['F_' num2str(ff) '_' num2str(ffs) '_' num2str(deltaF) '_' num2str(tm/1000) '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  F_' num2str(ff) '_' num2str(ffs) '_' num2str(deltaF) '_' num2str(tm/1000)
'_'num2str(SNR_dB)]);
   disp(['Signal only save as :          F_' num2str(ff) '_' num2str(ffs) '_' num2str(deltaF) '_' num2str(tm/1000)
'_s']);
   disp(['Directory:                    ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

```
%*************************************************************************
% frank.m
%
% Use:  Code to generated FRANK-coded signals
%
% Inputs:      Amplitud of the carrier signal
%                       Carrier signal frequency - f (Hz)
%             Sampling frequency - fs (HZ)
%             Desired SNR in dB
%             Number of phases
%             Number of cycles per phase
%
%
%Output:       In-phase (I) and Quadarture (Q) components of the signal
%             Plots
%
%May 1, 2002
%Maj Fernando Taboada - Venezuelan Army
%*************************************************************************

clear all;
clc;
disp('*******************************************');
disp('**************FRANK  CODE ******************');
disp('*******************************************');

%DEFAULT VARIABLES
A=1;                    % Amplitude of CW
f =1e3;                 % Carrier frequency
fs =7e3;                % Sample Rate
SNR_dB = 0;                %Signal to Noise Ratio
scale=30;               % Scaling for plotting time domain graphs
j=sqrt(-1);             % j
m=8;                    % Number of code phases
cpp = 1;                %Number of cycles per phase

% NEW INPUT
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitude of the carrier signal - A= %g.\n', A)
   fprintf('2. Carrier frequency - f (Hz) = %g.\n', f)
   fprintf('3. Sampling frequency - fs (Hz)= %g.\n', fs)
   fprintf('4. Signal to noise ratio - SNR_dB (dB) = %g.\n', SNR_dB)
   fprintf('5. Number of phase codes - m = %g.\n', m)
   fprintf('6. Number of cycles per phase - cpp = %g.\n', cpp)
   fprintf('7. No changes\n')
   disp(' ')
   option= input('Select a option: ');

   switch option
   case 1
      A=input('New amplitude of the carrier signal= ');
   case 2
```

```
      f=input('New carrier frequency (Hz) = ');
   case 3
      fs=input('New sampling frequency (Hz)= ');
   case 4
     SNR_dB=input('New signal to noise ratio (dB)= ');
   case 5
      m=input('New number of phase codes =');
   case 6
      cpp=input('New number of cycles per phase=');
   case 7
      newvar = 0;
   end
   clc;
end

SAR=ceil(fs/f);         % Sampling ratio
tb=1/(fs);              % Sampling period


N=m;
%Creating the phase matrix

   for i=1:m
      for j=1:m
         phi(i,j)=2*pi/N*(i-1)*(j-1);
      end
   end

% This section generates I & Q without Frank phase shift and I & Q with Phase shift.  The signals are generated
% five times by the outer loop.  The variable 'index' is used to generate a time vector for time domain plots.
% The signal is generated at seven samples per phase change.

index=0;
for p=1:5 %Generate the signal five times and store sequentially in corresponding vectors
   for i=1:m %Loop to shift phase
      for j=1:m
         for n=1:SAR*cpp                              %Loop to increment time for single phase value.
            I(index+1)=A*cos(2*pi*f*(n-1)*tb+phi(i,j)); %Calculate in phase component of signal with phase
shift
            IWO(index+1)=A*cos(2*pi*f*(n-1)*tb); % Calculate in phase component of signal without phase
shift
            Q(index+1)=A*sin(2*pi*f*(n-1)*tb+phi(i,j)); % Calculate quadrature component of signal with phase
shift
            QWO(index+1)=A*sin(2*pi*f*(n-1)*tb); %Calculate quadrature component of signal without phase
shift
            time(index+1)=index*tb; %time vector cumulation
            index = index +1;
         end
      end
   end
end

%Power Spectral Density for I with phase shift & with WGN with Signal to noise ratios (SNR) = [0,-5,5,10,-
10,-20]
%for loop makes calculations and plots for each value of SNR for WGN
[a,b]=size(I);
```

```
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power);%calculate noise at specified SNR
IN=I+noise;              %add noise to I with Frank phase shift
IPWON=I;                %I with phase shift without noise
QN=Q+noise;             %add noise to Q with Frank phase shift
QPWON=Q;                %Q with phase shift without noise

ff=floor(f/1e3);
ffs=floor(fs/1e3);
%*******************************************************
%PLOTS
%*******************************************************

disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
   disp(' ')
   %Plot Power Spectral Density for I without phase shift
   figurecount=1; %figurecount is plot index
   figure (figurecount); % open new figure for plot
   psd(IWO,[],fs); %Power Spectral Density of I without Phase shift
   title([' PSD of I without Phase Shift or Noise']);
   figurecount=figurecount+1; %increment figure count

   %time domain plot of in phase signal I without phase shift
   figure (figurecount); % open new figure for plot
   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot

   plot (time(1:floor(size(time,2)/scale)),IWO(1:floor(size(time,2)/scale)));
   title([ ' Time Domain of I without Phase Shift or Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1; %increment figure index

   %Power Spectral Density for I with phase shift
   figure (figurecount); %open new figure for plot
   psd(I,[],fs); %plot power spectral density of I with phase shift
   title([' PSD of I Phase Shift & no Noise']);
   figurecount=figurecount+1; %increment figure index
   %time domain plot of in phase signal I with phase shift
   figure (figurecount); %open new figure for plot

   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot
   plot (time(1:floor(size(time,2)/scale)),I(1:floor(size(time,2)/scale)));
   title([' Time Domain of I with Phase Shift & no Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1;%increment figure index
```

```
%Plot PSD and Time Domain of I+ Frank Phase + WGN and Time Domain of I + Frank Phase

figure (figurecount);% open new figure for plot
psd(IN,[],fs);%plot PSD for specified noise SNR
title([' PSD of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
figurecount=figurecount+1;%increment figure index

%plot time domain signal I with Frank phase shift and WGN at specified SNR
figure (figurecount);%open new figure for plot
plot(time(1:floor(size(time,2)/scale)),IN(1:floor(size(time,2)/scale)));
title([' Time Domain of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
xlabel('{\itTime - Seconds} ');
ylabel('Amplitude');
grid on;
figurecount=figurecount+1;%increment figure index

figure (figurecount);% open new figure for plot
plot(time(1:floor(size(time,2)/scale)),IPWON(1:floor(size(time,2)/scale)));
title([' Time Domain of I with Phase Shift witoutout Noise']);
xlabel('{\itTime - Seconds} ');
ylabel ('Amplitude');
grid on;
figurecount=figurecount+1;%increment figure index

% Now check to see if signal is correct by plotting phase shift alone and then determining phase shift from I+jQ.
% To determine phase shift, look at the phase angle of I+jQ at every 7th time interval.  Expect to see the Frank phase
% function plot repeated 5 times after unwrapping and detrending the phase angle.

figure(figurecount);%open new figure for plot
plot(phi);
title(['Frank Code Phase Shift']);
xlabel('i - index for phase change');
ylabel('Frank Phase Shift - Theta');
grid on;
figurecount=figurecount+1;%increment figure index


figure(figurecount);%open new figure for plot

%to concatenate the phase matrix in only 1-row matrix
nn=0;
for ii=1:N
   for jj=1:N
      nn=nn+1;
      phi2(nn)=phi(ii,jj);
   end
end

xx=0:length(phi2)-1;
plot(xx,phi2);grid
title(['Frank Code Phase Shift']);
xlabel('i - index for phase change');
ylabel('Frank Phase Shift - Theta');
figurecount=figurecount+1;%increment figure index
```

```
   %Now strip out points from I and Q to reconstruct phase shift.
   %I(1:SAR:floor(size(I,2)/5)) selects a data points with the phase values corresponding to the original phase calculation,;
   %by indexing SAR through the first one fifth of the vector computed by floor(size(I,)/5).  The vector was repeated five times.
   signal=I(1:SAR*cpp:size(I,2))+j*Q(1:SAR*cpp:size(I,2));

   phase_signal=angle(signal);%determine the angle from the complex signal

   % unwrap(I) corrects the radian phase angles in array I by adding multiples of ±2pi
   % when absolute jumps between consecutive array elements are greater than pi radians.
   unphase=unwrap(phase_signal);
   figure (figurecount);%open new figure for plot
   plot (unphase);
   title([' Frank Code Phase Shift from I+jQ ']);
   xlabel('i - index for phase change');
   ylabel('Frank Phase Shift - Theta');
   grid on;
   figurecount=figurecount+1;%increment figure index


else
   disp('Signal not plotted')
   fprintf('\n\n')
end

% This section generates the files for analysis

INP=IN';%transpose I with noise and Frank phase shift for text file
QNP=QN';%transpose Q with noise and Frank phase shift for text file
IPWONT=IPWON';%transpose I with phase without noise for text file
QPWONT=QPWON';%transpose Q with phase without noise for text file

% % save results in data files

I= INP(:,1);
Q=QNP(:,1);

II= IPWONT(:,1);
QQ=QPWONT(:,1);

disp(' ')
saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');

if (saveresult == 'Y') | (saveresult =='y')


   save(['FR_', num2str(ff), '_', num2str(ffs), '_', num2str(m), '_', num2str(cpp) ,'_', num2str(SNR_dB)],'I','Q');
   I=II;
   Q=QQ;
   save(['FR_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  FR_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_' num2str(SNR_dB)]);
   disp(['Signal only save as :        FR_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s']);
```

```
   disp(['Directory:                ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

```
%*******************************************************************************
% p1.m
%
% Use:  Code to generated P1-coded signals
%
% Inputs:      Amplitud of the carrier signal
%                       Carrier signal frequency - f (Hz)
%             Sampling frequency - fs (HZ)
%             Desired SNR in dB
%             Number of phases
%             Number of cycles per phase
%
%
%Output:       In-phase (I) and Quadarture (Q) components of the signal
%             Plots
%
%May 1, 2002
%Maj Fernando Taboada - Venezuelan Army
%*******************************************************************************

clear all;
clc;
disp('*****************************************');
disp('******************P1  CODE ******************');
disp('*****************************************');

%DEFAULT VARIABLES
A=1;                    % Amplitude of CW
f =1e3;                 % Carrier frequency
fs =7e3;                % Sample Rate
SNR_dB = 0;                 %Signal to Noise Ratio
scale=30;               % Scaling for plotting time domain graphs
j=sqrt(-1);             % j
m=8;                    % Number of code phases
cpp = 1;                %Number of cycles per phase

% NEW INPUT
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitude of the carrier signal - A= %g.\n', A)
   fprintf('2. Carrier frequency - f (Hz) = %g.\n', f)
   fprintf('3. Sampling frequency - fs (Hz)= %g.\n', fs)
   fprintf('4. Signal to noise ratio - SNR_dB (dB) = %g.\n', SNR_dB)
   fprintf('5. Number of phase codes - m = %g.\n', m)
   fprintf('6. Number of cycles per phase - cpp = %g.\n', cpp)
   fprintf('7. No changes\n')
   disp(' ')
   option= input('Select a option: ');

   switch option
   case 1
      A=input('New amplitude of the carrier signal= ');
   case 2
```

```
      f=input('New carrier frequency (Hz) = ');
   case 3
      fs=input('New sampling frequency (Hz)= ');
   case 4
     SNR_dB=input('New signal to noise ratio (dB)= ');
   case 5
      m=input('New number of phase codes =');
   case 6
      cpp=input('New number of cycles per phase=');
   case 7
      newvar = 0;
   end
   clc;
end

SAR=ceil(fs/f);         % Sampling ratio
tb=1/(fs);              % Sampling period
% Phase code for P4 from IEEE Internaltional Radar Conference Paper
% SIDELOBE REDUCTION TECHNIQUES FOR POLYPHASE PULSE COMPRESSSION CODES
% by P1 F. Kretshcmer and Laurence R. Welch.
%


%Creating the phase matrix
N=m;
   for i=1:m
      for j=1:m
         phi(i,j)= (-pi/N)*[N-(2*j-1)]*[(j-1)*N+(i-1)];
      end
   end


%
% This section generates I & Q without P4 phase shift and I & Q with Phase shift.  The signals are generated
% five times by the outer loop.  The variable 'index' is used to generate a time vector for time domain plots.
% The signal is generated at seven samples per phase change.

index=0;
for p=1:5 %Generate the signal five times and store sequentially in corresponding vectors
   for i=1:m %Loop to shift phase
      for j=1:m
         for n=1:SAR*cpp                              %Loop to increment time for single phase value.
            I(index+1)=A*cos(2*pi*f*(n-1)*tb+phi(i,j)); %Calculate in phase component of signal with phase
shift
            IWO(index+1)=A*cos(2*pi*f*(n-1)*tb); % Calculate in phase component of signal without phase
shift
            Q(index+1)=A*sin(2*pi*f*(n-1)*tb+phi(i,j)); % Calculate quadrature component of signal with phase
shift
            QWO(index+1)=A*sin(2*pi*f*(n-1)*tb); %Calculate quadrature component of signal without phase
shift
            time(index+1)=index*tb; %time vector cumulation
            index = index +1;
         end
      end
   end
end
```

```
%Power Spectral Density for I with phase shift & with WGN with Signal to noise ratios (SNR) = [0,-5,5,10,-
10,-20]
%for loop makes calculations and plots for each value of SNR for WGN
[a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power);%calculate noise at specified SNR
IN=I+noise;              %add noise to I with P4 phase shift
IPWON=I;                 %I with phase shift without noise
QN=Q+noise;              %add noise to Q with P4 phase shift
QPWON=Q;                 %Q with phase shift without noise


%******************************************************
%PLOTS
%******************************************************

disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
   disp(' ')
   %Plot Power Spectral Density for I without phase shift
   figurecount=1; %figurecount is plot index
   figure (figurecount); % open new figure for plot
   psd(IWO,[],fs); %Power Spectral Density of I without Phase shift
   title(['Fig #' num2str(figurecount) ' PSD of I without Phase Shift or Noise']);
   figurecount=figurecount+1; %increment figure count

   %time domain plot of in phase signal I without phase shift
   figure (figurecount); % open new figure for plot
   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot

   plot (time(1:floor(size(time,2)/scale)),IWO(1:floor(size(time,2)/scale)));
   title(['Fig #' num2str(figurecount) ' Time Domain of I without Phase Shift or Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1; %increment figure index

   %Power Spectral Density for I with phase shift
   figure (figurecount); %open new figure for plot
   psd(I,[],fs); %plot power spectral density of I with phase shift
   title(['Fig #' num2str(figurecount) ' PSD of I Phase Shift & no Noise']);
   figurecount=figurecount+1; %increment figure index
   %time domain plot of in phase signal I with phase shift
   figure (figurecount); %open new figure for plot

   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot
   plot (time(1:floor(size(time,2)/scale)),I(1:floor(size(time,2)/scale)));
   title(['Fig #' num2str(figurecount) ' Time Domain of I with Phase Shift & no Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
```

```
grid on;
figurecount=figurecount+1;%increment figure index


%Plot PSD and Time Domain of I+ P4 Phase + WGN and Time Domain of I + P4 Phase

figure (figurecount);% open new figure for plot
psd(IN,[],fs);%plot PSD for specified noise SNR
title(['Fig #' num2str(figurecount) ' PSD of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
figurecount=figurecount+1;%increment figure index

%plot time domain signal I with P4 phase shift and WGN at specified SNR
figure (figurecount);%open new figure for plot
plot(time(1:floor(size(time,2)/scale)),IN(1:floor(size(time,2)/scale)));
title(['Fig #' num2str(figurecount) ' Time Domain of I with Phase Shift & Noise SNR='
num2str(10*log10(SNR))]);
xlabel('{\itTime - Seconds} ');
ylabel('Amplitude');
grid on;
figurecount=figurecount+1;%increment figure index

figure (figurecount);% open new figure for plot
plot(time(1:floor(size(time,2)/scale)),IPWON(1:floor(size(time,2)/scale)));
title(['Fig #' num2str(figurecount) ' Time Domain of I with Phase Shift witoutout Noise']);
xlabel('{\itTime - Seconds} ');
ylabel ('Amplitude');
grid on;
figurecount=figurecount+1;%increment figure index

% Now check to see if signal is correct by plotting phase shift alone and then determining phase shift from
I+jQ.
% To determine phase shift, look at the phase angle of I+jQ at every 7th time interval.  Expect to see the P4
phase
% function plot repeated 5 times after unwrapping and detrending the phase angle.

figure(figurecount);%open new figure for plot
plot(phi);
title(['Fig #' num2str(figurecount) 'P1 Code Phase Shift']);
xlabel('i - index for phase change');
ylabel('P1 Phase Shift - Theta');
grid on;
figurecount=figurecount+1;%increment figure index


figure(figurecount);%open new figure for plot

%to concatenate the phase matrix in only 1-row matrix
nn=0;
for ii=1:N
   for jj=1:N
      nn=nn+1;
      phi2(nn)=phi(ii,jj);
   end
end

xx=0:length(phi2)-1;
```

```
   plot(xx,phi2);grid
   title(['Fig #' num2str(figurecount) 'P1 Code Phase Shift']);
   xlabel('i - index for phase change');
   ylabel('P1 Phase Shift - Theta');
   figurecount=figurecount+1;%increment figure index

   %Now strip out points from I and Q to reconstruct phase shift.
   %I(1:SAR:floor(size(I,2)/5)) selects a data points with the phase values corresponding to the original phase
calculation,;
   %by indexing SAR through the first one fifth of the vector computed by floor(size(I,)/5).  The vector was
repeated five times.
   signal=I(1:SAR*cpp:size(I,2))+j*Q(1:SAR*cpp:size(I,2));

   phase_signal=angle(signal);%determine the angle from the complex signal

   % unwrap(I) corrects the radian phase angles in array I by adding multiples of ±2pi
   % when absolute jumps between consecutive array elements are greater than pi radians.
   unphase=unwrap(phase_signal);
   figure (figurecount);%open new figure for plot
   plot (unphase);
   title(['Fig #' num2str(figurecount) ' P1 Code Phase Shift from I+jQ ']);
   xlabel('i - index for phase change');
   ylabel('P4 Phase Shift - Theta');
   grid on;
   figurecount=figurecount+1;%increment figure index


else
   disp('Signal not plotted')
   fprintf('\n\n')
end

% This section generates the files for analysis

INP=IN';%transpose I with noise and P4 phase shift for text file
QNP=QN';%transpose Q with noise and P4 phase shift for text file
IPWONT=IPWON';%transpose I with phase without noise for text file
QPWONT=QPWON';%transpose Q with phase without noise for text file

% % save results in data files

I= INP(:,1);
Q=QNP(:,1);

II= IPWONT(:,1);
QQ=QPWONT(:,1);

disp(' ')
saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');

if (saveresult == 'Y') | (saveresult =='y')
   ff=floor(f/1e3);
   ffs=floor(fs/1e3);
   save(['P1_', num2str(ff), '_', num2str(ffs), '_', num2str(m), '_',num2str(cpp),'_',num2str(SNR_dB)],'I','Q');
   I=II;
   Q=QQ;
```

```
   save(['P1_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  P1_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_'
num2str(SNR_dB)]);
   disp(['Signal only save as :       P1_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s']);
   disp(['Directory:                  ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

```
%*************************************************************************
% p2.m
%
% Use:  Code to generated P2-coded signals
%
% Inputs:      Amplitud of the carrier signal
%                       Carrier signal frequency - f (Hz)
%             Sampling frequency - fs (HZ)
%             Desired SNR in dB
%             Number of phases
%             Number of cycles per phase
%
%
%Output:       In-phase (I) and Quadarture (Q) components of the signal
%             Plots
%
%July 18, 2002
%Maj Fernando Taboada - Venezuelan Army
%*************************************************************************

clear all;
clc;
disp('*******************************************');
disp('******************P2  CODE ******************');
disp('*******************************************');

%DEFAULT VARIABLES
A=1;                    % Amplitude of CW
f =1e3;                 % Carrier frequency
fs =7e3;                % Sample Rate
SNR_dB = 0;                 %Signal to Noise Ratio
scale=30;               % Scaling for plotting time domain graphs
j=sqrt(-1);             % j
m=8;                    % Number of code phases
cpp = 1;                %Number of cycles per phase

% NEW INPUT
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitude of the carrier signal - A= %g.\n', A)
   fprintf('2. Carrier frequency - f (Hz) = %g.\n', f)
   fprintf('3. Sampling frequency - fs (Hz)= %g.\n', fs)
   fprintf('4. Signal to noise ratio - SNR_dB (dB) = %g.\n', SNR_dB)
   fprintf('5. Number of phase codes - m = %g.\n', m)
   fprintf('6. Number of cycles per phase - cpp = %g.\n', cpp)
   fprintf('7. No changes\n')
   disp(' ')
   option= input('Select a option: ');

   switch option
   case 1
      A=input('New amplitude of the carrier signal= ');
   case 2
```

```
      f=input('New carrier frequency (Hz) = ');
   case 3
      fs=input('New sampling frequency (Hz)= ');
   case 4
     SNR_dB=input('New signal to noise ratio (dB)= ');
   case 5
      m=input('New number of phase codes =');
   case 6
      cpp=input('New number of cycles per phase=');
   case 7
      newvar = 0;
   end
   clc;
end

SAR=ceil(fs/f);         % Sampling ratio
tb=1/(fs);              % Sampling period
% Phase code for P2 from IEEE Internaltional Radar Conference Paper
% SIDELOBE REDUCTION TECHNIQUES FOR POLYPHASE PULSE COMPRESSSION CODES
% by Frank F. Kretshcmer and Laurence R. Welch.
%

N=m;

%Creating the phase matrix

for i=1:m
   for j=1:m
      phi(i,j)= (-pi/(2*N))*[2*i-1-N]*[2*j-1-N];
   end
end



%
% This section generates I & Q without P2 phase shift and I & Q with Phase shift.  The signals are generated
% five times by the outer loop.  The variable 'index' is used to generate a time vector for time domain plots.
% The signal is generated at seven samples per phase change.

index=0;
for p=1:5 %Generate the signal five times and store sequentially in corresponding vectors
   for i=1:m %Loop to shift phase
      for j=1:m
         for n=1:SAR*cpp                              %Loop to increment time for single phase value.
            I(index+1)=A*cos(2*pi*f*(n-1)*tb+phi(i,j)); %Calculate in phase component of signal with phase
shift
            IWO(index+1)=A*cos(2*pi*f*(n-1)*tb); % Calculate in phase component of signal without phase
shift
            Q(index+1)=A*sin(2*pi*f*(n-1)*tb+phi(i,j)); % Calculate quadrature component of signal with phase
shift
            QWO(index+1)=A*sin(2*pi*f*(n-1)*tb); %Calculate quadrature component of signal without phase
shift
            time(index+1)=index*tb; %time vector cumulation
            index = index +1;
         end
      end
```

```
   end
end

%Power Spectral Density for I with phase shift & with WGN with Signal to noise ratios (SNR) = [0,-5,5,10,-
10,-20]
%for loop makes calculations and plots for each value of SNR for WGN
[a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power);%calculate noise at specified SNR
IN=I+noise;              %add noise to I with P2 phase shift
IPWON=I;              %I with phase shift without noise
QN=Q+noise;           %add noise to Q with P2 phase shift
QPWON=Q;              %Q with phase shift without noise


%*******************************************************
%PLOTS
%*******************************************************

disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
   disp(' ')
   %Plot Power Spectral Density for I without phase shift
   figurecount=1; %figurecount is plot index
   figure (figurecount); % open new figure for plot
   psd(IWO,[],fs); %Power Spectral Density of I without Phase shift
   title([' PSD of I without Phase Shift or Noise']);
   figurecount=figurecount+1; %increment figure count

   %time domain plot of in phase signal I without phase shift
   figure (figurecount); % open new figure for plot
   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot

   plot (time(1:floor(size(time,2)/scale)),IWO(1:floor(size(time,2)/scale)));
   title([' Time Domain of I without Phase Shift or Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1; %increment figure index

   %Power Spectral Density for I with phase shift
   figure (figurecount); %open new figure for plot
   psd(I,[],fs); %plot power spectral density of I with phase shift
   title([' PSD of I Phase Shift & no Noise']);
   figurecount=figurecount+1; %increment figure index
   %time domain plot of in phase signal I with phase shift
   figure (figurecount); %open new figure for plot

   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot
   plot (time(1:floor(size(time,2)/scale)),I(1:floor(size(time,2)/scale)));
   title([' Time Domain of I with Phase Shift & no Noise']);
```

```
xlabel('{\itTime - Seconds} ');
ylabel('Amplitude');
grid on;
figurecount=figurecount+1;%increment figure index


%Plot PSD and Time Domain of I+ P2 Phase + WGN and Time Domain of I + P2 Phase

figure (figurecount);% open new figure for plot
psd(IN,[],fs);%plot PSD for specified noise SNR
title([' PSD of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
figurecount=figurecount+1;%increment figure index

%plot time domain signal I with P2 phase shift and WGN at specified SNR
figure (figurecount);%open new figure for plot
plot(time(1:floor(size(time,2)/scale)),IN(1:floor(size(time,2)/scale)));
title([' Time Domain of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
xlabel('{\itTime - Seconds} ');
ylabel('Amplitude');
grid on;
figurecount=figurecount+1;%increment figure index

figure (figurecount);% open new figure for plot
plot(time(1:floor(size(time,2)/scale)),IPWON(1:floor(size(time,2)/scale)));
title([' Time Domain of I with Phase Shift witoutout Noise']);
xlabel('{\itTime - Seconds} ');
ylabel ('Amplitude');
grid on;
figurecount=figurecount+1;%increment figure index


figure(figurecount);%open new figure for plot

%to concatenate the phase matrix in only 1-row matrix
nn=0;
for ii=1:N
   for jj=1:N
      nn=nn+1;
      phi2(nn)=phi(ii,jj);
   end
end
xx=0:length(phi2)-1;
plot(xx,phi2);grid
title(['P2 Code Phase Shift']);
xlabel('i - index for phase change');
ylabel('P2 Phase Shift - Theta');
figurecount=figurecount+1;%increment figure index

%Now strip out points from I and Q to reconstruct phase shift.
%I(1:SAR:floor(size(I,2)/5)) selects a data points with the phase values corresponding to the original phase calculation,;
%by indexing SAR through the first one fifth of the vector computed by floor(size(I,)/5).  The vector was repeated five times.
signal=I(1:SAR*cpp:size(I,2))+j*Q(1:SAR*cpp:size(I,2));

phase_signal=angle(signal);%determine the angle from the complex signal
```

```
   % unwrap(I) corrects the radian phase angles in array I by adding multiples of ±2pi
   % when absolute jumps between consecutive array elements are greater than pi radians.
   unphase=unwrap(phase_signal);
   figure (figurecount);%open new figure for plot
   plot (unphase);
   title([' P2 Code Phase Shift from I+jQ ']);
   xlabel('i - index for phase change');
   ylabel('P2 Phase Shift - Theta');
   grid on;



else
   disp('Signal not plotted')
   fprintf('\n\n')
end

% This section generates the files for analysis

INP=IN';%transpose I with noise and P2 phase shift for text file
QNP=QN';%transpose Q with noise and P2 phase shift for text file
IPWONT=IPWON';%transpose I with phase without noise for text file
QPWONT=QPWON';%transpose Q with phase without noise for text file

% % save results in data files
I= INP(:,1);
Q=QNP(:,1);

II= IPWONT(:,1);
QQ=QPWONT(:,1);

disp(' ')
saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');

if (saveresult == 'Y') | (saveresult =='y')
   ff=floor(f/1e3);
   ffs=floor(fs/1e3);
    save(['P2_', num2str(ff), '_', num2str(ffs), '_', num2str(m), '_',num2str(cpp),'_',num2str(SNR_dB)],'I','Q');
   I=II;
   Q=QQ;
   save(['P2_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  P2_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_'
num2str(SNR_dB)]);
   disp(['Signal only save as :  P2_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s']);
   disp(['Directory:  ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

```
%*******************************************************************************
% p3.m
%
% Use:  Code to generated P3-coded signals
%
% Inputs:      Amplitud of the carrier signal
%                       Carrier signal frequency - f (Hz)
%             Sampling frequency - fs (HZ)
%             Desired SNR in dB
%             Number of phases
%             Number of cycles per phase
%
%
%Output:       In-phase (I) and Quadarture (Q) components of the signal
%             Plots
%
%May 1, 2002
%Maj Fernando Taboada - Venezuelan Army
%*******************************************************************************

clear all;
clc;
disp('*******************************************');
disp('******************P3  CODE ******************');
disp('*******************************************');

%DEFAULT VARIABLES
A=1;                    % Amplitude of CW
f =1e3;                 % Carrier frequency
fs =7e3;                % Sample Rate
SNR_dB = 0;                %Signal to Noise Ratio
scale=30;               % Scaling for plotting time domain graphs
j=sqrt(-1);             % j
m=64;                   % Number of code phases
cpp = 1;                %Number of cycles per phase

% NEW INPUT
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitude of the carrier signal - A= %g.\n', A)
   fprintf('2. Carrier frequency - f (Hz) = %g.\n', f)
   fprintf('3. Sampling frequency - fs (Hz)= %g.\n', fs)
   fprintf('4. Signal to noise ratio - SNR_dB (dB) = %g.\n', SNR_dB)
   fprintf('5. Number of phase codes - m = %g.\n', m)
   fprintf('6. Number of cycles per phase - cpp = %g.\n', cpp)
   fprintf('7. No changes\n')
   disp(' ')
   option= input('Select a option: ');

   switch option
   case 1
      A=input('New amplitude of the carrier signal= ');
   case 2
```

```
      f=input('New carrier frequency (Hz) = ');
   case 3
      fs=input('New sampling frequency (Hz)= ');
   case 4
     SNR_dB=input('New signal to noise ratio (dB)= ');
   case 5
      m=input('New number of phase codes =');
   case 6
      cpp=input('New number of cycles per phase=');
   case 7
      newvar = 0;
   end
   clc;
end

SAR=ceil(fs/f);         % Sampling ratio
tb=1/(fs);              % Sampling period

for k = 1:m
   phase(k)=(pi/m)*(k-1)*(k-1); %Compute the P3 Phase
end


%
% This section generates I & Q without P3 phase shift and I & Q with Phase shift.  The signals are generated
% five times by the outer loop.  The variable 'index' is used to generate a time vector for time domain plots.
% The signal is generated at seven samples per phase change.

index=0;
for p=1:5 %Generate the signal five times and store sequentially in corresponding vectors
for l=1:m %Loop to shift phase
   for n=1:SAR*cpp %Loop to increment time for single phase value.
    I(index+1)=A*cos(2*pi*f*(n-1)*tb+phase(l)); %Calculate in phase component of signal with phase shift
   IWO(index+1)=A*cos(2*pi*f*(n-1)*tb); % Calculate in phase component of signal without phase shift
   Q(index+1)=A*sin(2*pi*f*(n-1)*tb+phase(l)); % Calculate quadrature component of signal with phase shift
   QWO(index+1)=A*sin(2*pi*f*(n-1)*tb); %Calculate quadrature component of signal without phase shift
   time(index+1)=index*tb; %time vector cumulation
   index = index +1;
   end
end
end

%Power Spectral Density for I with phase shift & with WGN with Signal to noise ratios (SNR) = [0,-5,5,10,-
10,-20]
%for loop makes calculations and plots for each value of SNR for WGN
[a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power);%calculate noise at specified SNR
IN=I+noise;             %add noise to I with P3 phase shift
IPWON=I;               %I with phase shift without noise
QN=Q+noise;            %add noise to Q with P3 phase shift
QPWON=Q;               %Q with phase shift without noise


%*****************************************************
```

```
%PLOTS
%**************************************************

disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
   disp(' ')
   %Plot Power Spectral Density for I without phase shift
   figurecount=1; %figurecount is plot index
   figure (figurecount); % open new figure for plot
   psd(IWO,[],fs); %Power Spectral Density of I without Phase shift
   title([' PSD of I without Phase Shift or Noise']);
   figurecount=figurecount+1; %increment figure count

   %time domain plot of in phase signal I without phase shift
   figure (figurecount); % open new figure for plot
   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot

   plot (time(1:floor(size(time,2)/scale)),IWO(1:floor(size(time,2)/scale)));
   title([' Time Domain of I without Phase Shift or Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1; %increment figure index

   %Power Spectral Density for I with phase shift
   figure (figurecount); %open new figure for plot
   psd(I,[],fs); %plot power spectral density of I with phase shift
   title([' PSD of I Phase Shift & no Noise']);
   figurecount=figurecount+1; %increment figure index
   %time domain plot of in phase signal I with phase shift
   figure (figurecount); %open new figure for plot

   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot
   plot (time(1:floor(size(time,2)/scale)),I(1:floor(size(time,2)/scale)));
   title([' Time Domain of I with Phase Shift & no Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1;%increment figure index


   %Plot PSD and Time Domain of I+ P3 Phase + WGN and Time Domain of I + P1 Phase

   figure (figurecount);% open new figure for plot
   psd(IN,[],fs);%plot PSD for specified noise SNR
   title([' PSD of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
   figurecount=figurecount+1;%increment figure index

   %plot time domain signal I with P1 phase shift and WGN at specified SNR
   figure (figurecount);%open new figure for plot
   plot(time(1:floor(size(time,2)/scale)),IN(1:floor(size(time,2)/scale)));
   title([' Time Domain of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
```

```
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1;%increment figure index

   figure (figurecount);% open new figure for plot
   plot(time(1:floor(size(time,2)/scale)),IPWON(1:floor(size(time,2)/scale)));
   title([' Time Domain of I with Phase Shift witoutout Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel ('Amplitude');
   grid on;
   figurecount=figurecount+1;%increment figure index

   % Now check to see if signal is correct by plotting phase shift alone and then determining phase shift from
I+jQ.
   % To determine phase shift, look at the phase angle of I+jQ at every 7th time interval.  Expect to see the P3
phase
   % function plot repeated 5 times after unwrapping and detrending the phase angle.

   xx=0:length(phase)-1;
   figure(figurecount);%open new figure for plot
   plot(xx,phase);
   title(['P3 Code Phase Shift']);
   xlabel('i - index for phase change');
   ylabel('Frank Phase Shift - Theta');
   grid on;
   figurecount=figurecount+1;%increment figure index



   %Now strip out points from I and Q to reconstruct phase shift.
   %I(1:SAR:floor(size(I,2)/5)) selects a data points with the phase values corresponding to the original phase
calculation,;
   %by indexing SAR through the first one fifth of the vector computed by floor(size(I,)/5).  The vector was
repeated five times.
   signal=I(1:SAR*cpp:size(I,2))+j*Q(1:SAR*cpp:size(I,2));

   phase_signal=angle(signal);%determine the angle from the complex signal

   % unwrap(I) corrects the radian phase angles in array I by adding multiples of ±2pi
   % when absolute jumps between consecutive array elements are greater than pi radians.
   unphase=unwrap(phase_signal);
   figure (figurecount);%open new figure for plot
   plot (unphase);
   title([' P3 Code Phase Shift from I+jQ ']);
   xlabel('i - index for phase change');
   ylabel('P3 Phase Shift - Theta');
   grid on;
   figurecount=figurecount+1;%increment figure index


else
   disp('Signal not plotted')
   fprintf('\n\n')
end
```

```
% This section generates the files for analysis

INP=IN';%transpose I with noise and P3 phase shift for text file
QNP=QN';%transpose Q with noise and P3 phase shift for text file
IPWONT=IPWON';%transpose I with phase without noise for text file
QPWONT=QPWON';%transpose Q with phase without noise for text file

% % save results in data files

I= INP(:,1);
Q=QNP(:,1);

II= IPWONT(:,1);
QQ=QPWONT(:,1);

disp(' ')
saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');

if (saveresult == 'Y') | (saveresult =='y')
   ff=floor(f/1e3);
   ffs=floor(fs/1e3);
    save(['P3_', num2str(ff), '_', num2str(ffs), '_', num2str(m), '_',num2str(cpp),'_',num2str(SNR_dB)],'I','Q');
   I=II;
   Q=QQ;
   save(['P3_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  P3_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_'
num2str(SNR_dB)]);
   disp(['Signal only save as :       P3_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s']);
   disp(['Directory:                  ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

```
%*************************************************************************
% p4.m
%
% Use:  Code to generated P4-coded signals
%
% Inputs:      Amplitud of the carrier signal
%                     Carrier signal frequency - f (Hz)
%            Sampling frequency - fs (HZ)
%            Desired SNR in dB
%            Number of phases
%            Number of cycles per phase
%
%
%Output:      In-phase (I) and Quadarture (Q) components of the signal
%            Plots
%
%January 22, 2002
%Maj Fernando Taboada - Venezuelan Army
%*************************************************************************

clear all;
clc;
disp('*******************************************');
disp('************POLIPHASE CODE (P4)*************');
disp('*******************************************');

%DEFAULT VARIABLES
A=1;                    % Amplitude of CW
f =1e3;                 % Carrier frequency
fs =7e3;                % Sample Rate
SNR_dB = 0;             %Signal to Noise Ratio
scale=30;               % Scaling for plotting time domain graphs
j=sqrt(-1);             % j
m=64;                   % Number of code phases
cpp = 1;                %Number of cycles per phase

% NEW INPUT
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitude of the carrier signal - A= %g.\n', A)
   fprintf('2. Carrier frequency - f (Hz) = %g.\n', f)
   fprintf('3. Sampling frequency - fs (Hz)= %g.\n', fs)
   fprintf('4. Signal to noise ratio - SNR_dB (dB) = %g.\n', SNR_dB)
   fprintf('5. Number of phase codes - m = %g.\n', m)
   fprintf('6. Number of cycles per phase - cpp = %g.\n', cpp)
   fprintf('7. No changes\n')
   disp(' ')
   option= input('Select a option: ');

   switch option
   case 1
      A=input('New amplitude of the carrier signal= ');
   case 2
```

```
      f=input('New carrier frequency (Hz) = ');
   case 3
      fs=input('New sampling frequency (Hz)= ');
   case 4
     SNR_dB=input('New signal to noise ratio (dB)= ');
   case 5
      m=input('New number of phase codes =');
   case 6
      cpp=input('New number of cycles per phase=');
   case 7
      newvar = 0;
   end
   clc;
end

SAR=ceil(fs/f);         % Sampling ratio
tb=1/(fs);              % Sampling period
% Phase code for P4 from IEEE Internaltional Radar Conference Paper
% SIDELOBE REDUCTION TECHNIQUES FOR POLYPHASE PULSE COMPRESSSION CODES
% by Frank F. Kretshcmer and Laurence R. Welch.
%

for k = 1:m
   phase(k)=((pi/m)*(k-1)^2)-(pi*(k-1)); %Compute the P4 Phase
end
%
% This section generates I & Q without P4 phase shift and I & Q with Phase shift.  The signals are generated
% five times by the outer loop.  The variable 'index' is used to generate a time vector for time domain plots.
% The signal is generated at seven samples per phase change.

index=0;
for p=1:5 %Generate the signal five times and store sequentially in corresponding vectors
for l=1:m %Loop to shift phase
   for n=1:SAR*cpp %Loop to increment time for single phase value.
    I(index+1)=A*cos(2*pi*f*(n-1)*tb+phase(l)); %Calculate in phase component of signal with phase shift
   IWO(index+1)=A*cos(2*pi*f*(n-1)*tb); % Calculate in phase component of signal without phase shift
   Q(index+1)=A*sin(2*pi*f*(n-1)*tb+phase(l)); % Calculate quadrature component of signal with phase shift
   QWO(index+1)=A*sin(2*pi*f*(n-1)*tb); %Calculate quadrature component of signal without phase shift
   time(index+1)=index*tb; %time vector cumulation
   index = index +1;
   end
end
end

%Power Spectral Density for I with phase shift & with WGN with Signal to noise ratios (SNR) = [0,-5,5,10,-
10,-20]
%for loop makes calculations and plots for each value of SNR for WGN
[a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power);%calculate noise at specified SNR
IN=I+noise;             %add noise to I with P4 phase shift
IPWON=I;               %I with phase shift without noise
QN=Q+noise;             %add noise to Q with P4 phase shift
QPWON=Q;               %Q with phase shift without noise
```

```
%*****************************************************
%PLOTS
%*****************************************************

disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
   disp(' ')
   %Plot Power Spectral Density for I without phase shift
   figurecount=1; %figurecount is plot index
   figure (figurecount); % open new figure for plot
   psd(IWO,[],fs); %Power Spectral Density of I without Phase shift
   title([' PSD of I without Phase Shift or Noise']);
   figurecount=figurecount+1; %increment figure count

   %time domain plot of in phase signal I without phase shift
   figure (figurecount); % open new figure for plot
   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot

   plot (time(1:floor(size(time,2)/scale)),IWO(1:floor(size(time,2)/scale)));
   title([' Time Domain of I without Phase Shift or Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1; %increment figure index

   %Power Spectral Density for I with phase shift
   figure (figurecount); %open new figure for plot
   psd(I,[],fs); %plot power spectral density of I with phase shift
   title([' PSD of I Phase Shift & no Noise']);
   figurecount=figurecount+1; %increment figure index
   %time domain plot of in phase signal I with phase shift
   figure (figurecount); %open new figure for plot

   % plot small portion of time domain signal I so that data will fit meaningfully in figure.
   %floor(size(time,2)/scale) selects a small sample of the vectors to plot
   plot (time(1:floor(size(time,2)/scale)),I(1:floor(size(time,2)/scale)));
   title([' Time Domain of I with Phase Shift & no Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1;%increment figure index


   %Plot PSD and Time Domain of I+ P4 Phase + WGN and Time Domain of I + P4 Phase

   figure (figurecount);% open new figure for plot
   psd(IN,[],fs);%plot PSD for specified noise SNR
   title([' PSD of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
   figurecount=figurecount+1;%increment figure index

   %plot time domain signal I with P4 phase shift and WGN at specified SNR
   figure (figurecount);%open new figure for plot
```

```
   plot(time(1:floor(size(time,2)/scale)),IN(1:floor(size(time,2)/scale)));
   title([' Time Domain of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
   xlabel('{\itTime - Seconds} ');
   ylabel('Amplitude');
   grid on;
   figurecount=figurecount+1;%increment figure index

   figure (figurecount);% open new figure for plot
   plot(time(1:floor(size(time,2)/scale)),IPWON(1:floor(size(time,2)/scale)));
   title([' Time Domain of I with Phase Shift witoutout Noise']);
   xlabel('{\itTime - Seconds} ');
   ylabel ('Amplitude');
   grid on;
   figurecount=figurecount+1;%increment figure index

   % Now check to see if signal is correct by plotting phase shift alone and then determining phase shift from
I+jQ.
   % To determine phase shift, look at the phase angle of I+jQ at every 7th time interval.  Expect to see the P4
phase
   % function plot repeated 5 times after unwrapping and detrending the phase angle.

   xx=0:length(phase)-1;
   figure(figurecount);%open new figure for plot
   plot(xx,phase);
   title(['P4 Phase Shift']);
   xlabel('i - index for phase change');
   ylabel('P4 Phase Shift - Theta');
   grid on;
   figurecount=figurecount+1;%increment figure index
   %Now strip out points from I and Q to reconstruct phase shift.
   %I(1:SAR:floor(size(I,2)/5)) selects a data points with the phase values corresponding to the original phase
calculation,;
   %by indexing SAR through the first one fifth of the vector computed by floor(size(I,)/5).  The vector was
repeated five times.
   signal=I(1:SAR*cpp:size(I,2))+j*Q(1:SAR*cpp:size(I,2));

   phase_signal=angle(signal);%determine the angle from the complex signal

   % unwrap(I) corrects the radian phase angles in array I by adding multiples of ±2pi
   % when absolute jumps between consecutive array elements are greater than pi radians.
   unphase=unwrap(phase_signal);
   figure (figurecount);%open new figure for plot
   plot (unphase);
   title([' P4 Phase Shift from I+jQ ']);
   xlabel('i - index for phase change');
   ylabel('P4 Phase Shift - Theta');
   grid on;
   figurecount=figurecount+1;%increment figure index


else
   disp('Signal not plotted')
   fprintf('\n\n')
end

% This section generates the files for analysis
```

```
INP=IN';%transpose I with noise and P4 phase shift for text file
QNP=QN';%transpose Q with noise and P4 phase shift for text file
IPWONT=IPWON';%transpose I with phase without noise for text file
QPWONT=QPWON';%transpose Q with phase without noise for text file

% % save results in data files

I= INP(:,1);
Q=QNP(:,1);

II= IPWONT(:,1);
QQ=QPWONT(:,1);

disp(' ')
saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');

if (saveresult == 'Y') | (saveresult =='y')
   ff=floor(f/1e3);
   ffs=floor(fs/1e3);
    save(['P4_', num2str(ff), '_', num2str(ffs), '_', num2str(m), '_',num2str(cpp),'_',num2str(SNR_dB)],'I','Q');
   I=II;
   Q=QQ;
   save(['P4_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  P4_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_'
num2str(SNR_dB)]);
   disp(['Signal only save as :       P4_' num2str(ff) '_' num2str(ffs) '_' num2str(m) '_' num2str(cpp) '_s']);
   disp(['Directory:                  ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

```
%*******************************************************************************
% costas.m
%
% Use:  Code to generated Costas-coded signals
%
% Inputs:      Amplitud of the carrier signal
%                       Sampling frequency - fs (HZ)
%             Desired SNR in dB
%             Costas sequence
%             Number of cycles per phase
%
%
%Output:       In-phase (I) and Quadarture (Q) components of the signal
%             Plots
%
%August 2, 2002
%Maj Fernando Taboada - Venezuelan Army
%*******************************************************************************

clear all;
clc;
disp('*********************************************');
disp('***************COSTAS CODE ****************');
disp('*********************************************');

%DEFAULT VARIABLES
A=1;                    % Amplitude of CW
fs =15e3;                % Sample Rate
SNR_dB = 0;                %Signal to Noise Ratio
cpf=10;                 %Cycles per frequency (> 10)
scale=30;                % Scaling for plotting time domain graphs
j=sqrt(-1);              % j


% NEW INPUT
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH PARAMETER DO YOU WANT TO SET ?  ')
   disp(' ')
   fprintf('1. Amplitude of frequencies - A= %g.\n', A)
   fprintf('2. Sampling frequency - fs (Hz)= %g.\n', fs)
   fprintf('3. Signal to noise ratio - SNR_dB (dB) = %g.\n', SNR_dB)
   fprintf('4. Cycles per frequency  = %g.\n', cpf)
   fprintf('5. No changes\n')
   disp(' ')
   option= input('Select a option: ');

   switch option
   case 1
      A=input('New amplitude of the carrier signal= ');
   case 2
      fs=input('New sampling frequency (Hz)= ');
   case 3
     SNR_dB=input('New signal to noise ratio (dB)= ');
   case 4
```

```
      cpf=input('New number of cycles per frequency = ');
   case 5
      newvar = 0;
   end
   clc;
end
% FREQUENCY CHOICES
newvar = 1;
while newvar == 1;
   disp(' ')
   disp('WHICH FREQUENCY WOULD YOU LIKE TO USE ?  ')
   disp(' ')
   disp('1.       4, 7, 1, 6, 5, 2, 3 ');
   disp('2.       2, 6, 3, 8, 7, 5, 1 ');
   disp(' ')
   optioN= input('Select an option: ');

freq=[4 7 1 6 5 2 3;
     2 6 3 8 7 5 1]*1000

switch optioN
   case 1
      seq=freq(1,:);
      fs=15e3;

   case 2
      seq=freq(2,:);
      fs=17e3;

   end
   newvar=0;
   clc;
end


minimum=min(seq);
SAR=ceil(fs/minimum);         % Sampling ratio
tb=1/(fs);                    % Sampling period


% This section generates I & Q without COSTAS phase shift and I & Q with Phase shift.  The signals are
generated
% five times by the outer loop.  The variable 'index' is used to generate a time vector for time domain plots.
% The signal is generated at seven samples per phase change.

index=0;
%for p=1:5 %Generate the signal five times and store sequentially in corresponding vectors
   for xx=1:7
      for n=1:SAR*cpf

            I(index+1)=A*cos(2*pi*seq(xx)*(n-1)*tb);

            Q(index+1)=A*sin(2*pi*seq(xx)*(n-1)*tb);

            time(index+1)=index*tb; %time vector cumulation
```

```
            index = index +1;
        end
    end

    %end


%Power Spectral Density for I with phase shift & with WGN with Signal to noise ratios
%for loop makes calculations and plots for each value of SNR for WGN
[a,b]=size(I);
SNR=10^(SNR_dB/10);
power=10*log10(A^2/(2*SNR));%calculate SNR in dB for WGN function
noise=wgn(a,b,power);%calculate noise at specified SNR
IN=I+noise;              %add noise to I with COSTAS phase shift
IPWON=I;                %I with phase shift without noise
QN=Q+noise;             %add noise to Q with COSTAS phase shift
QPWON=Q;                %Q with phase shift without noise


%******************************************************
%PLOTS
%******************************************************

disp(' ')
plt = input('Do you want to generate plots of the signal (Y/y or N/n) ?','s');
disp(' ')
if (plt == 'Y') | (plt =='y')
    disp(' ')

    figurecount=1;
    %Power Spectral Density for I with phase shift
    figure (figurecount); %open new figure for plot
    psd(I,[],fs); %plot power spectral density of I with phase shift
    title(['Fig #' num2str(figurecount) ' PSD of I Phase Shift & no Noise']);

    figurecount=figurecount+1; %increment figure index



    %Plot PSD and Time Domain of I+ COSTAS Phase + WGN and Time Domain of I

    figure (figurecount);% open new figure for plot
    psd(IN,[],fs);%plot PSD for specified noise SNR
    title(['Fig #' num2str(figurecount) ' PSD of I with Phase Shift & Noise SNR=' num2str(10*log10(SNR))]);
    figurecount=figurecount+1;%increment figure index



else
    disp('Signal not plotted')
    fprintf('\n\n')
end

% This section generates the files for analysis

INP=IN';%transpose I with noise and COSTAS phase shift for text file
```

```
QNP=QN';%transpose Q with noise and COSTAS phase shift for text file
IPWONT=IPWON';%transpose I with phase without noise for text file
QPWONT=QPWON';%transpose Q with phase without noise for text file

% % save results in data files

I= INP(:,1);
Q=QNP(:,1);

II= IPWONT(:,1);
QQ=QPWONT(:,1);

disp(' ')
saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');

if (saveresult == 'Y') | (saveresult =='y')
   ff=7;
   ffs=floor(fs/1e3);
   save(['C_' num2str(optioN) '_' num2str(ffs) '_' num2str(cpf) '_'num2str(SNR_dB)],'I','Q');
   I=II;
   Q=QQ;
   save(['C_' num2str(optioN) '_' num2str(ffs) '_' num2str(cpf) '_s'],'I','Q');
   disp(' ');
   disp(['Signal and noise save as :  C_' num2str(optioN) '_' num2str(ffs) '_' num2str(cpf)
'_'num2str(SNR_dB)]);
   disp(['Signal only save as :  C_' num2str(optioN) '_' num2str(ffs) '_' num2str(cpf) '_s']);
   disp(['Directory:  ' num2str(cd)]);
else
   disp(' ')
   disp('Signal not saved')
   fprintf('\n\n')
end
```

```
%*******************************************************************************
% test_signal.m
%
% Use:  Code to generated a single tone test signals
%
% Inputs:       Amplitud of the carrier signal
%                       Carrier signal frequency - f (Hz)
%             Sampling frequency - fs (HZ)
%
%
%
%
%
%Output:       In-phase (I) and Quadarture (Q) components of the signal
%             Plots
%
%January 18, 2002
%Maj Fernando Taboada - Venezuelan Army
%*******************************************************************************


clear all;
clc;
disp('*********************************************');
disp('*****************TEST SIGNAL*****************');
disp('*********************************************');

%MAIN MENU
disp(' ')
disp('WHAT KIND OF TEST SIGNAL DO YOU WANT TO GENERATE ?  ')
disp(' ')
disp('1. Test signal with a single frequency ');
disp('2. Test signal with two frequencies ');
disp('3. Go back to LPI Generator ');
disp(' ')
option= input('Select a option: ');

%AMPLITUDE IN VOLTS
A = 1;

%MENU FOR SINGLE FREQUENCY TEST SIGNAL
switch option
   case 1
      disp(' ');
      disp('You have selected a Test signal with a single frequency');
      disp(' ');
            f1 = input('Enter carrier frequency - f1(Hz): ');
            fs = input('Enter sampling frequency - fs (Hz): ');
            T=1/fs;
            t=0:T:100*T;
            s = A*cos(2*pi*f1*t) + j*A*sin(2*pi*f1*t);
            figure
            hold;
            plot(t(1:10),real(s(1:10)));
            plot(t(1:10),imag(s(1:10)),'r');grid;
```

```
        title(['Test Signal S(t) = Cos[2(pi)(f1)t] + jSin[2(pi)(f1)t], frequency (f1) =' num2str(f1) ' Sampling
Frequency(fs)=' num2str(fs) ]);
        xlabel('{\itTime - Seconds} ');
        ylabel('Amplitude');
        legend('Real Part', 'Imag. Part');
        hold;
        figure;
        psd(s,[],fs)
        title('Power Spectral Density');

      disp(' ')
      saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');
      if (saveresult == 'Y') | (saveresult =='y')
        I = real(s);
        Q = imag(s);

        ff=floor(f1/1e3);
        ffs=floor(fs/1e3);
        save(['T_' num2str(ff) '_' num2str(ffs) '_1_s'],'I','Q');
        disp('The signals has been saved');
        disp(' ');
        disp(['Signal only save as :      T_' num2str(ff) '_' num2str(ffs) '_1_s']);
        disp(['Directory:                 ' num2str(cd)]);
      else
        disp(' ')
        disp('Signal not saved')
        fprintf('\n\n')
      end


  %MENU FOR TWO FREQUENCY TEST SIGNAL
  case 2
    disp('You have selected Test signal with two frequencies');
    disp(' ');
        f1 = input('Enter carrier frequency - f1 (Hz): ');
        f2 = input('Enter carrier frequency - f2 (Hz): ');
        fs = input('Enter sampling frequency - fs (Hz): ');
        T=1/fs;
        t=0:T:100*T;
        s = cos(2*pi*f1*t) + j*sin(2*pi*f1*t) + cos(2*pi*f2*t) + j*sin(2*pi*f2*t);
        figure
        hold;
        plot(t(1:10),real(s(1:10)));
        plot(t(1:10),imag(s(1:10)),'r');grid;
        title(['Test Signal S(t) = Cos[2(pi)(f1)t] + jSin[2(pi)(f1)t] + Cos[2(pi)(f2)t] + jSin[2(pi)(f2)t], f1 ='
num2str(f1) ' f2=' num2str(f2) ' fs=' num2str(fs) ]);
        xlabel('{\itTime - Seconds} ');
        ylabel('Amplitude');
        legend('Real Part', 'Imag. Part');
        hold;
        figure;
        psd(s,[],fs)
        title('Power Spectral Density');
      disp(' ')
      saveresult = input('Do you want to save the new signal (Y/y or N/n) ?','s');
      if (saveresult == 'Y') | (saveresult =='y')
```

```
        I = real(s);
        Q = imag(s);
         ff=floor(f1/1e3);
         fff=floor(f2/1e3);
         ffs=floor(fs/1e3);
         save(['T_' num2str(ff)num2str(fff) '_' num2str(ffs) '_2_s'],'I','Q');
         disp('The signals has been saved');
         disp(' ');
         disp(['Signal only save as :        T_' num2str(ff)num2str(fff) '_' num2str(ffs) '_2_s']);
         disp(['Directory:                    ' num2str(cd)]);
      else
         disp(' ')
         disp('Signal not saved')
         fprintf('\n\n')
      end


case 3
   LPI_signal_generator;
end
```

APPENDIX B. MATLAB® PROGRAM FOR PARALLEL FILTER AND HOS

```
%*************************************************************************
% hos_gui_1.m
%
% Use:  This program creates a GUI for the implementation of parallel filter
%      array and HOS
%
% Inputs:      File name
%              Directory
%              Sampling frequency
%              Number of filters in bank
%
%Output:       Four (04) plots:
%              1. After parallel filter arrays
%              2. After HOS
%              3. Amplitude-filter view
%              4. Amplitude-frequency view
%
%July 5, 2002
%Maj Fernando Taboada - Venezuelan Army
%*************************************************************************


function varargout = HOS_gui_1(varargin)
% HOS_GUI_1 Application M-file for HOS_gui_1.fig
%    FIG = HOS_GUI_1 launch HOS_gui_1 GUI.
%    HOS_GUI_1('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 23-Apr-2002 14:17:40

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Use system color scheme for figure:
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

        try
                if (nargout)
                        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
```

```
                else
                        feval(varargin{:}); % FEVAL switchyard
                end
        catch
                disp(lasterr);
        end

end

% --------------------------------------------------------------------
function varargout = edit1_Callback(h, eventdata, handles, varargin)
name = get(h,'String');
handles.edit1 = name; % Save file name in global variable handles.edit1 (filename = handles.edit1)

if isstr(name)
else
   errordlg('Filename: Please enter a valid file name (B_1_7_7_s is default)', 'Bad input', 'modal')
end

guidata(h,handles);



% --------------------------------------------------------------------
function varargout = edit2_Callback(h, eventdata, handles, varargin)


d = get(h,'String');
handles.edit3 = d; % Save file name in global variable handles.edit1 (filename = handles.edit1)


guidata(h,handles);


% --------------------------------------------------------------------
function varargout = edit3_Callback(h, eventdata, handles, varargin)
fs = str2double(get(h,'String'));
handles.edit2 = fs;% Save sampling frequency in global variable handles.edit2 (fs = handles.edit2)

if isnan(fs)
   errordlg('Sampling Frequency: Please enter a numeric value (Hz)', 'Bad input', 'modal')
end

guidata(h,handles);



% --------------------------------------------------------------------
function varargout = popupmenu1_Callback(h, eventdata, handles, varargin)
val = get(h,'Value');
switch val
case 1
   handles.popupmenu3 = 32;% Save frequency resolution in global variable handles.popupmenu3 (df =
handles.popupmenu3)

case 2
```

```
   handles.popupmenu3 = 64;% Save frequency resolution in global variable handles.popupmenu3 (df = handles.popupmenu3)

case 3
   handles.popupmenu3 = 128;% Save frequency resolution in global variable handles.popupmenu3 (df = handles.popupmenu3)

end
% More options may be added following the same pattern.

guidata(h,handles);



% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)

taboada_HOS_gui




%*************************************************************************
% taboada_HOS.m
%
% Use:  Implementation of parallel filter
%       array and HOS
%
% Inputs:       None (parameters entered in GUI)
%
%
%
%
%Output:        Four (04) plots:
%               1. After parallel filter arrays
%               2. After HOS
%               3. Amplitude-filter view
%               4. Amplitude-frequency view
%
%July 5, 2002
%Maj Fernando Taboada - Venezuelan Army
%*************************************************************************

%This code was written to realize the detection outlined in the
%paper "On Detection Using Filter Banks and HOS"
% by Farook Sattar and Goran Salonsson.

clc;
disp('Executing.....')

%load data from file and GUI
L=  handles.popupmenu3;
fs=handles.edit2;
d=handles.edit3;
cd(d);
```

```
name= handles.edit1;
load (name)



%Now Build the real input
X=I-j*Q;
%Flip input around the hard way
for i=1:length(X)-1;
  x(1,i)=X(i);
end;
T=1/fs;
f=0:fs/length(x):fs-fs/length(x);  %build the f vector
t=0:T:(length(x)-1)*T;

%Filter the input 2 times, one time with regular filter, then again with
% filter that is Hilbert transform of the first filter. Next step is to
% compare and combine the outputs.
% Build first filter
w=2*pi*f;
wp=2*pi*fs/(2*L);
qp=.707;
k=1;
s=j*w;
num=k*wp^2;
den=s.^2+(wp/qp).*s+wp^2;
hf=num./den;
fi=0;
ht=ifft(hf);

%Now set up input file for processing by
%   1) zero pad input out to a lenth of input + length of filter -1
%      This is done in order to prevent circular convolution
%   2) Then take fft of input to bring it to frequency domain.
zpad=zeros(1,length(ht)-1);
xpad=[x,zpad];
xfd=fft(xpad);
seqlen=length(xpad);    %set the length of the input sequence
%now pad the filter with zeros to prevent circular convolution
%This section defines the window and the N used for the
%cumulant calculation
K=2;
windowlen=2*K+1;
Wn=(blackman(windowlen))';      %get window function for Blackman window of length 2*K+1.

flagval=0;

for index=0:L-1
          %Bring filter to time domain
          %Convert to bandpass filter with pass band = fi
  %Make sure to use zero pad to avoid circular convolution
  ht2=ht.*exp(j*2*pi*fi*t);
  htpad=[ht2,zpad];
  hcosine=real(htpad);      %cosine filter
  hsine=imag(htpad);        %sine filter
```

```
        % Get first output
        %Bring input signal and filter to Freq domain and filter
hfbp=fft(hcosine);
%hbp(index+1,:)=hfbp;       %Remove comment if want to look at filter bank
        yfd1=xfd.*hfbp;
        %Go back to the time domain
yt1=ifft(yfd1);
y1(index+1,:)=(yt1);   % Save output before run cummulant.
Y1(index+1,:)=fft((yt1));

%Now code 3rd order cumulator using
% the formula C(L1,L2,k)=(1/(S2-S1+1)*sum(zk(n)zk(n+L1)zk(n+L2)))
%      where zk(n) = z(n)w(n-k) where w(n-k) is taken as a Hamming Window.
        %Equation 13 in the paper says that the 3rd order cumulant of the kth output is just
% row(k)=C3(0,0;k)-C3(-1,1;k) where C3 indicates cumulant
%Find C3(0,0;k) and C3(-1,1;k)
%First find C3(0,0,;k);
for k=2:seqlen;
  L1=0;
        L2=0;
  S1=max([k-K,k-K-L1,k-K-L2]);
        S2=min([k+K,k+K-L1,k+K-L2]);
  Zsubn=0;
  C3_0_0=0;
  for n=S1:S2;%Now find zsubk(n)=z(n)*w(n-k), z(n) is input w(n-k) is window I built
    %Basically have to run the window function on each z(n) value.
    %Then sum up the values to find the zsubk I need.
    %Plan is to get z(n-2), z(n-1), z(n), z(n+1), z(n+2)
    %    run the window function on them, sum them and then take the cube
    pindex=1;
    for p=n-2:1:n+2;  %Get 5 values of input to running hamming window on
      if p <= 0;     %This little if statement makes sure I stay within index
              Zsubp(pindex)=0;
          elseif p>seqlen;
          Zsubp(pindex)=0;
        else
        Zsubp(pindex)=(yt1(p));    %Pick off the position I need
        end;
      pindex=pindex+1;          %gota keep my index going
    end;
    %Now run window function on that values I just found.
    Zsubp=(Zsubp.*Wn);
    Zsubk=(sum(Zsubp))^3;%sum values after window function and take cube
    C3_0_0=C3_0_0+Zsubk;      %Cummulate my cumulant
  end;
  C3_0_0=C3_0_0*1/(S2-S1+1);   %Finish up cumulant calculation


  %Now find the second part, C3(-1,1;k)
          %Now find zsubk(n)=z(n)*w(n-k) z(n) is input w(n-k) is window I built
  %Basically have to run the window function on each z(n) value.
  %   Then sum up the values to find the zsubk I need.
  %   Plan is to get z(n-2), z(n-1), z(n), z(n+1), z(n+2)
  %   run the window function on them, sum them and then take the cube
  %    then find zsubk(n-1) and zsubk(n+1) is a similar manner.
```

```
L1=-1;
L2=1;
S1=max([k-K,k-K-L1,k-K-L2]);
S2=min([k+K,k+K-L1,k+K-L2]);
C3_1_1=0;
for n=S1:S2;
  %Now find zsub(n) first;
  pindex=1;
  for p=n-2:1:n+2;  %Get 5 values of input to running hamming window on
    if p <= 0;     %This little if statement makes sure I stay within index
            Zsubp(pindex)=0;
        elseif p>seqlen;
        Zsubp(pindex)=0;
      else
      Zsubp(pindex)=(yt1(p));    %Pick off the position I need
      end;
    pindex=pindex+1;          %gota keep my index going
  end;
  %Now run window function on that values I just found.
  Zsubp=(Zsubp.*Wn);
  Zsubk=sum(Zsubp);
  %Now find zsub(n-1) next:
  pindex=1;
  for p=n-3:1:n+1;  %Get 5 values of input to running hamming window on
    if p <= 0;     %This little if statement makes sure I stay within index
            Zsubp(pindex)=0;
        elseif p>seqlen;
        Zsubp(pindex)=0;
      else
      Zsubp(pindex)=(yt1(p));    %Pick off the position I need
      end;
    pindex=pindex+1;          %gota keep my index going
  end;
  %Now run window function on that values I just found.
  Zsubp=(Zsubp.*Wn);
  Zsubkm1=sum(Zsubp);
  %Now find zsub(n+1);
  pindex=1;
  for p=n-1:1:n+3;  %Get 5 values of input to running hamming window on
    if p <= 0;     %This little if statement makes sure I stay within index
            Zsubp(pindex)=0;
        elseif p>seqlen;
        Zsubp(pindex)=0;
      else
      Zsubp(pindex)=(yt1(p));    %Pick off the position I need
      end;
    pindex=pindex+1;          %gota keep my index going
  end;
  %Now run window function on that values I just found.
  Zsubp=(Zsubp.*Wn);
  Zsubkp1=sum(Zsubp);
  Zprod=Zsubk*Zsubkm1*Zsubkp1;
  C3_1_1=C3_1_1+Zprod;

  if (Zprod==0)
    flagval=flagval+1;
```

```
    end;

  end;
  ThirdCum=C3_0_0-C3_1_1;
  %Now save this value in position n of the output stream;
  ycumout(k)=ThirdCum;

end; %end of for k:seqlenth;
YcumI(index+1,:)=ycumout;
%Get second output
hhfd=fft(hsine);       %goto Freq domain.
%Get the output
yfd2=xfd.*hhfd;
        %Go back to the time domain
        yt2=ifft(yfd2);
        y2(index+1,:)=(yt2);
% Y2(index+1,:)=fft((yt2));   %Remove comment if want to see Freq domain
  %Now code 3rd order cumulator using
% the formula C(L1,L2,k)=(1/(S2-S1+1)*sum(zk(n)zk(n+L1)zk(n+L2)))
%      where zk(n) = z(n)w(n-k) where w(n-k) is taken as a Hamming Window.
        %Equation 13 in the paper says that the 3rd order cumulant of the kth output is just
% row(k)=C3(0,0;k)-C3(-1,1;k) where C3 indicates cumulant
%Find C3(0,0;k) and C3(-1,1;k)
%First find C3(0,0,;k);
for k=2:seqlen;
  L1=0;
        L2=0;
  S1=max([k-K,k-K-L1,k-K-L2]);
        S2=min([k+K,k+K-L1,k+K-L2]);
  Zsubn=0;
  C3_0_0=0;
  for n=S1:S2;%Now find zsubk(n)=z(n)*w(n-k), z(n) is input w(n-k) is window I built
    %Basically have to run the window function on each z(n) value.
    %Then sum up the values to find the zsubk I need.
    %Plan is to get z(n-2), z(n-1), z(n), z(n+1), z(n+2)
    %    run the window function on them, sum them and then take the cube
    pindex=1;
    for p=n-2:1:n+2;  %Get 5 values of input to running hamming window on
      if p <= 0;     %This little if statement makes sure I stay within index
            Zsubp(pindex)=0;
         elseif p>seqlen;
          Zsubp(pindex)=0;
        else
        Zsubp(pindex)=(yt2(p));    %Pick off the position I need
        end;
      pindex=pindex+1;          %gota keep my index going
    end;
    %Now run window function on that values I just found.
    Zsubp=(Zsubp.*Wn);
    Zsubk=(sum(Zsubp))^3;     %sum values after window function and take cube
    C3_0_0=C3_0_0+Zsubk;      %Cummulate my cumulant
  end;
  C3_0_0=C3_0_0*1/(S2-S1+1);   %Finish up cumulant calculation

  %Now find the second part, C3(-1,1;k)
          %Now find zsubk(n)=z(n)*w(n-k) z(n) is input w(n-k) is window I built
```

```
%Basically have to run the window function on each z(n) value.
%   Then sum up the values to find the zsubk I need.
%   Plan is to get z(n-2), z(n-1), z(n), z(n+1), z(n+2)
%   run the window function on them, sum them and then take the cube
%    then find zsubk(n-1) and zsubk(n+1) is a similar manner.
L1=-1;
L2=1;
S1=max([k-K,k-K-L1,k-K-L2]);
S2=min([k+K,k+K-L1,k+K-L2]);
C3_1_1=0;
for n=S1:S2;
  %Now find zsub(n) first;
  pindex=1;
  for p=n-2:1:n+2;  %Get 5 values of input to running hamming window on
    if p <= 0;     %This little if statement makes sure I stay within index
            Zsubp(pindex)=0;
        elseif p>seqlen;
       Zsubp(pindex)=0;
      else
      Zsubp(pindex)=(yt2(p));    %Pick off the position I need
      end;
    pindex=pindex+1;         %gota keep my index going
  end;
  %Now run window function on that values I just found.
  Zsubp=(Zsubp.*Wn);
  Zsubk=sum(Zsubp);
  %Now find zsub(n-1) next:
  pindex=1;
  for p=n-3:1:n+1;  %Get 5 values of input to running hamming window on
    if p <= 0;     %This little if statement makes sure I stay within index
            Zsubp(pindex)=0;
        elseif p>seqlen;
       Zsubp(pindex)=0;
      else
      Zsubp(pindex)=(yt2(p));    %Pick off the position I need
      end;
    pindex=pindex+1;         %gota keep my index going
  end;
  %Now run window function on that values I just found.
  Zsubp=(Zsubp.*Wn);
  Zsubkm1=sum(Zsubp);
  %Now find zsub(n+1);
  pindex=1;
  for p=n-1:1:n+3;  %Get 5 values of input to running hamming window on
    if p <= 0;     %This little if statement makes sure I stay within index
            Zsubp(pindex)=0;
        elseif p>seqlen;
       Zsubp(pindex)=0;
      else
      Zsubp(pindex)=(yt2(p));    %Pick off the position I need
      end;
    pindex=pindex+1;         %gota keep my index going
  end;
  %Now run window function on that values I just found.
  Zsubp=(Zsubp.*Wn);
  Zsubkp1=sum(Zsubp);
```

```
        Zprod=Zsubk*Zsubkm1*Zsubkp1;
        C3_1_1=C3_1_1+Zprod;

        if (Zprod==0)
          flagval=flagval+1;
        end;

      end;
      C3_1_1=C3_1_1*1/(S2-S1+1);   %Finish up cumulant calculation
      ThirdCum=C3_0_0-C3_1_1;
      %Now save this value in position n of the output stream;
      ycumout2(k)=ThirdCum;
           end;
   YcumQ(index+1,:)=ycumout2;
   fii(index+1)=fi;       %Build the subfilter frequency vector
   fi=fi+fs/(2*(L));


end;  %end of for index:to sequenlen.
y=y1+j*y2;
yy=abs(y1)+abs(y2);     %to see what happens after filter bank
% Y=Y1+Y2;              %Remove comment if want freq domain output
Filterwidth = fs/(2*L);   %Subfilters Bandwidth
Ycum=abs(YcumQ)+abs(YcumI);

%Energy Factor

for A=1:L
    for B= 1:length(y)
        if Ycum(A,B) < 0.025
            Ycum_F(A,B)=0;
        else
            Ycum_F(A,B)=Ycum(A,B);
        end
    end
end

tiempo=0:1/fs:length(Ycum)/fs-1/fs;
% %**************************************
% %PLOTS
% %**************************************

%After filter Banks and Before Cumulants
figure(1)
mesh(abs(yy)); view(0,90); grid minor;
title('Signal after Filter Bank (No Cumulants) ');
xlabel('Samples');
ylabel(['Filter - Bandwidth: ', num2str(Filterwidth),' Hz']);

%After Cumulants
figure(2)
mesh(tiempo,fii,abs(Ycum)); view(0,90); grid minor;
title('Signal with Cumulator ');
xlabel('Time (sec)');
ylabel(['Frequency (Hz) - Bandwidth: ', num2str(Filterwidth),' Hz']);
```

```
%Filter vs Amplitude
figure(3)
mesh(abs(Ycum)); grid minor
view(90,0);
title('Signal with Cumulator in Frequency Domain');
zlabel('Amplitude');
ylabel(['Filter - Bandwidth: ', num2str(Filterwidth),' Hz']);

%Plot subfilters frequency vs. amplitude
figure(4)
plot(fii,abs(Ycum)); grid minor
title('Signal with Cumulator in Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

cd('h:\thesis\code\Data\GUI')
```

GLOSSARY OF ACRONYMS

ARM	Anti-Radiation Missile
BPSK	Binary Phase Shift-Keying
CW	Continuous Wave
EA	Electronic Attack
ES	Electronic Support
FFT	Fast Fourier Transform
FMCW	Frequency Modulated Continuous Wave
FSK	Frequency Shift Keying
HOS	Higher Order Statistics
I	In- phase component
LPI	Low Probability of Intercept
PAF	Periodic Ambiguity Function
PDF	Probability Density Function
PSK	Phase Shift Keying
PSD	Power Spectral Density
Q	Quadrature component
RSR	Random-signal Radars
RWR	Radar Warning Receiver
SNR	Signal-to-noise Ratio
WGN	White Gaussian Noise

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] . Pace, Phillip, "LPI Radar", class notes, EC Network Centric Radar Electronic Warfare Techniques and Systems for International Students, Naval Postgraduate School, January, 2002.

[2] . Adamy, Dave, "Advances in Signal Processing for Electronic Warfare", September 1995, retrieved from http://www.jedefense.com/default.asp?journalid=4.htm, accessed on 07/08/02.

[3] . Klein, Lawrence A., "Millimeter-Wave and Infrared Multisensor Design and Signal Processing", Artech House, Inc, 1997.

[4] . Currier, Walter, "Digital Receivers", Naval Postgraduate School, EC4700, June 2001.

[5] . Arcasoy, C.C., "On Cross-Ambiguity Properties of Welch-Costas Arrays", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 30, No. 4, October 1994.

[6] . Lewis, Bernard, "Aspects of Radar Signal Processing", Norwood, MA, Artech House, INC., 1986.

[7] . Levanon, Nadav, "PAF of CW Signals with Perfect Periodic Autocorrelation", *IEEE Transactions on Aerospace and Electronic System*, Vol. 28 No.2, April 1992.

[8] . Lima, Antonio, "Analysis of Low Probability of Intercept (LPI) Signals using Cyclostationary Processing", Master of Science in System Engineering , Naval Postgraduate School, Monterey CA, September 2002.

[9] . Sattar, Farook,"On Detection Using Filter Banks and Higher Order Statistics", *IEEE Transactions on Aerospace and Electronic System*, Vol. 36 No.4, October 2000.

[10] . Sattar, Farrok, "Nonparametric Waveform Estimation Using Filter Banks," Department of Electric and Computer Science, Lund University, Sweden, December 1993.

[11] . Mendel, Jerry M., "Signal Processing with Higher-Order Statistics", *IEEE Signal Processing Magazine*, July 1993.

[12] . Mendel, Jerry M., "Tutorial on Higher-Order Statistics (Spectra) in Signal Processing and System Theory: Theoretical Results and Some Applications", *Proceedings of the IEEE,* Vol. 79, No. 3, March 1991.

[13] . Wirth, W. D., "Polyphase Coded CW Radar", IEEE International Conference of Radar, Paris 1989.

[14] . McLaughlin, S. and Stogioglou, A, "Introducing Higher-Order Statistics (HOS) for the Detection of Non-linearities", http://www.amsta.leeds.ac.uk/Applied/news.dir/issues2/hos_intro.html. Accessed on March 1, 2002.

[15] . Haspel, Mitch, "Polyphase Signals for Radar", IEEE International Conference on Communication Systems, Singapore, 1990.

[16] . Donohoe, J. Patrick, "The Ambiguity Properties of FSK/PSK Signals", Radar Conference, 1990., Record of the IEEE 1990 International , 1990 Page(s): 268 –273

[17] . Schrick, Gerd, "Interception of LPI Radar Signals", Proceedings of the IEEE International Radar Conference,24-28 April 1989, Paris.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5121

3. Dan C. Boger, Chairman, Code 37
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. IW, EW Curricular Officer, Code 37
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Prof. Phillip Pace, Code EC
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Prof. Herschel H. Loomis Jr, Code EC
   Naval Postgraduate School
   Monterey, CA 93943-5121

7. Maj. Fernando Taboada
   6105 Raleigh st. apt. 322
   Orlando, FL 32835

8. Venezuelan Defense Attaché
   Venezuelan Embassy
   2409 California St. NW
   Washington DC 20008